Universitat de Girona
Escola Politècnica Superior

Treball final de grau

Estudi: Grau en Tecnologies Industrials

Títol: Improvement of the carbon composite skin of the fuselage of an aircraft by means of optimization algorithms to define laminate stacking sequences

Document: Memòria i annexos

Alumne: Axel García García

Tutor: Josep Costa Balanzat

Departament: Física

Àrea: Materials

Convocatòria (mes/any): Juny 2015

INDEX

ANNEXES

1. INTRODUCTION

1.1. Background

Nowadays the use of composite materials is really common in industries because of its low weight in comparison with its structural benefits. In aeronautic industries, there is a constant interest in lightening all of its components. Especially on civil aircraft fuselages which are done with skin and reinforcements. One of the design's objectives of this skin is to avoid buckling.

1.2. Objective

Our objective is to create a laminate that can accomplish the fuselage's skin's requirements reducing its weight and preserving its mechanical integrity.

1.3. Scope and project specifications

Programming with MATLAB we will develop an optimization algorithm based on the Ant Colony Optimization which explores the different solutions satisfying the mechanical's necessities and reducing the weight of the baseline.

We will use the knowledge of laminate theory to study different ways to define laminates. And we will use the ant colony optimization modified to find different solutions, changing the laminate thickness.

2. INTRODUCTION TO COMPOSITE MATERIALS

2.1. What a composite material is?

A composite material is the combination of two, or more than two, materials that improves its base properties working together.

A composite material must accomplish the following conditions:

- Every material which is part of the laminate has to have a proportion higher than a 5%.
- Every constituent have different properties to the others and to the resulting composite.
- Every material has to be recognizable at microscopic/macroscopic level. It means that it have to keep its integrity.

Composite materials have two components: base material (matrix) and another material which improve its properties (reinforcement).

The matrix's function is to support, protect and transfer the loads. Matrix can be done with three kinds of materials: ceramic, metallic or polymer.

Reinforcement's function is to improve the properties of the matrix component and to support the loads which are applied to the material. Usually the reinforcement material is stiffer and more resistant than the matrix. There are different dispositions of reinforcement, it generally appears like fibre or particles which dimensions are less than 500 µm.

The properties which we might want to improve combining these materials are:

- Stiffness
- Resistance
- Density
- Wear resistance
- Tenacity
- Thermal insulation
- Acoustic insulation
- Thermal conductivity
- Corrosion resistance
- Etc.

2.2. How to class it?

Composite materials are classified according to matrix's material. So we can class them as:

- **Polymeric Matrix Composite (PMC):** It's the most common composite. It's also known as FRP (*Fibre Reinforced Polymers*), and it's based on a polymeric matrix reinforced with a great variety of fibres (glass, carbon or aramid).
- **Metallic Matrix Composite (MMC):** Its matrix is done with metallic materials like aluminium, titanium. The reinforcement is usually silicon carbide or other hard ceramics. These kinds of composites are used on automobile industry.
- **Ceramic Matrix Composite (CMC):** They use ceramic materials as a matrix and are reinforced with short fibres, particles or *whiskersed* silicon carbide. It's used on high temperatures applications

3. CLASSIC LAMINATE THEORY[1]

3.1. Introduction

Composite laminates are built by different stacked layers. Every layer can have a different orientation than the following or not. A really important fact in order to design the laminate is that layers are really resistant on the fibre direction. So it's very interesting to control every layer's orientation.

3.2. Laminate stresses

There are nine different stresses and some of them are dependents on the others. In order to represent them, we used the following notation (Figure 1): σ_{ij} where *i* mean the plain's normal direction where the stress is acting and *j* is the vector's direction of the applied traction.

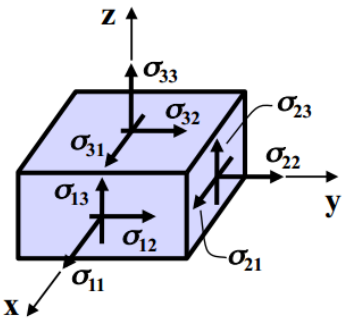


Figure 1 Stress representation

These stresses can be represented in a matrix as it's showed in Equation 1.

$$[\sigma] = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{12} & \sigma_{22} & \sigma_{23} \\ \sigma_{13} & \sigma_{23} & \sigma_{33} \end{bmatrix} \quad \text{(Eq.1)}$$

Equation 1 Stress matrix

There are two kinds of stresses: Normal and shear stresses. When *i=j* stress is normal, the others are shear stresses.

3.2.1. Hooke's law's constitutive equation

We can connect the material deformations and stresses as we can see in Equation 2.

$$\sigma_{ij} = C_{ijkl}\,\varepsilon_{kl} \quad \text{(Eq.2)}$$

Equation 2 Hooke's law's constitutive equation

C_{ijk} is the material stiffness on this direction.

3.3. Stiffness matrix

The performance of the material against different stresses or strains is defined by the stiffness matrix. In our case, we will assume that each ply is a transversely isotropic material. Therefore we will need 4 engineering constants to define our material properties: E_{11}, E_{22}, G_{12} i ν_{12}.

3.3.1. Local stiffness matrix

Stiffness matrix [C] also known as [Q] will be the same on every layer because our laminate will be completely made with the same composite material.

The easiest and quickest way of calculating Q matrixes is by means of Equation 3. That's because the compliance matrix, [S], is directly depending on the material properties as we can see in Equation 4.

$$[Q] = [S]^{-1} \tag{Eq.3}$$

Equation 3 Q and S relation

Compliance matrix [S] is defined as:

$$[S] = \begin{bmatrix} \frac{1}{E_{11}} & \frac{\nu_{12}}{E_{11}} & 0 \\ \frac{\nu_{12}}{E_{11}} & \frac{1}{E_{22}} & 0 \\ 0 & 0 & \frac{1}{G_{12}} \end{bmatrix} \tag{Eq.4}$$

Equation 4 Compliance matrix

But these matrixes only take in account material properties, in order to represent the layer orientation we need to calculate another matrix called [Qb]. The way to calculate this matrix is using matrix [S] and the rotation matrix, [T], which will modify the stiffness matrix depending on the layer orientation. This matrix is defined in Equation 5.

3.3.2. Transformation matrix

This matrix transforms the local properties of the layer to the global axes (because of the material rotation). For every new orientation we'll have a new [T] matrix.

We are working with in plane stress, σ_{11}, σ_{22} i σ_{12}, so our matrix will only contemplate this 3 stresses.

$$[T]=\begin{bmatrix} \cos^2\theta & \sin^2\theta & 2\cos\theta\sin\theta \\ \sin^2\theta & \cos^2\theta & -2\cos\theta\sin\theta \\ -\cos\theta\sin\theta & \cos\theta\sin\theta & \cos^2\theta-\sin^2\theta \end{bmatrix} \quad \text{(Eq.5)}$$

Equation 5 Transformation matrix [T]

Representation of θ can be found in Figure 3.

3.3.3. Q global matrix calculus

The easiest way to calculate Q global matrix is by inverting [Sb]. That's because the compliance matrix [S], as we said before, depends directly on

material properties and it can be transformed in [Sb] by applying Equation 7. After that calculation, finding Q becomes trivial using Equation 6.

$$[Qb]=[Sb]^{-1} \quad \text{(Eq.6)}$$

Equation 6 Qb calculus knowing Sb

Where:

$$[Sb]=[T_\gamma]^{-1}[S][T] \quad \text{(Eq.7)}$$

Equation 7 Sb calculus.

Where:

$$[T_\gamma]=\left([T]^{-1}\right)^t \quad \text{(Eq.8)}$$

Equation 8 T_γ definition

3.4. Stresses and strains calculus

Calculate the stress generated by a strain and vice versa is a simple calculation if you have previously defined the [Qb]. Equation 9 is the link between both of these concepts.

$$[\sigma]=[Q][\varepsilon] \quad \text{(Eq.9)}$$

Equation 9 Relation between stress and deformation

3.5. Layer numeration

First of all, on a laminate we have to set a global coordinate system. The first step is to define the origin of our system; it'll be at the middle of the laminate. Then the Z orientation will be perpendicular to our laminate and positive upwards. Then X have to coincide with the fibres that are orientate at 0º and Y perpendicular to it as it's showed at Figure 2.

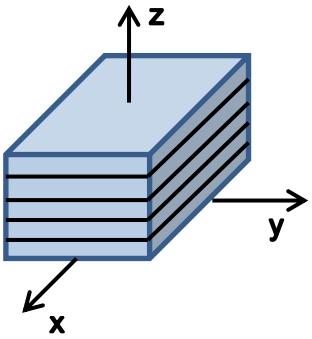


Figure 2 Global coordinate system

After defining our global coordinate system, we have to define every layer's local coordinate system. Its coordinates will be X_1, X_2 i X_3. Every orientation is showed at Figure 3 in green.

X_1 have the same direction as the fibres, X_2 is perpendicular to the fibre orientation and X_3 perpendicular to layer plane. The origin of the system will be again in the middle of the layer.

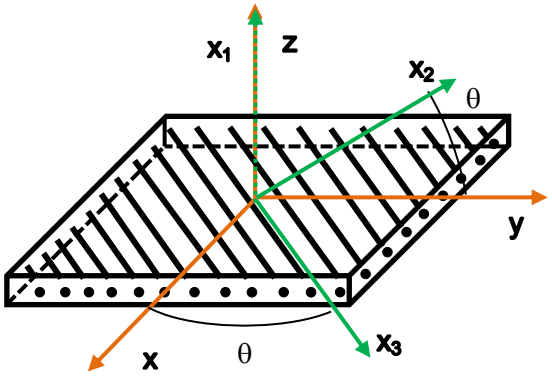


Figure 3 Local coordinate system

Next step is to establish our Z coordinate for every layer. Layers are numbered from outside to inside. We have N layers, then the top layer is Z_N and the bottom layer is Z_0, as we can see in Figure 4.

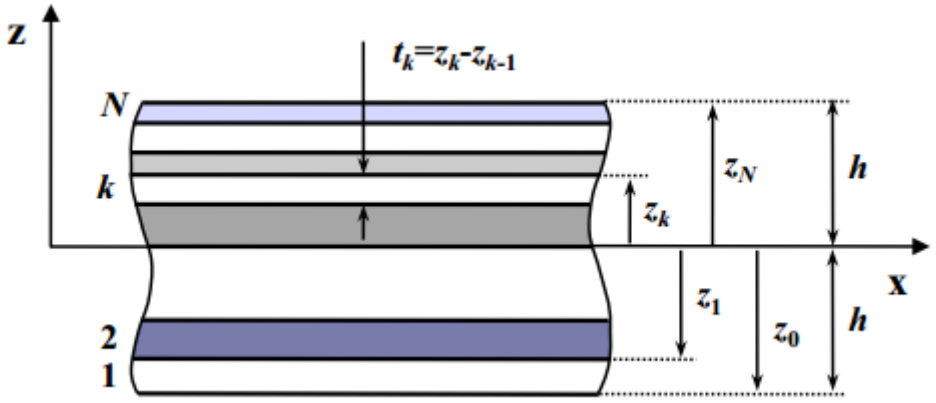


Figure 4 Layer's surface numeration

3.6. Kirchhoff theory

The deformation of composite materials has its peculiarities, they follow the Kirchhoff theory of thin plies under traction and buckling (Figure 5). The plane union keeps straight when laminate is deformed. Another characteristic is that mid planes normal have always the same length. This fact is because γ_{zy} and γ_{zx} are equal to 0. And z movements are related to x and y, so ε_{33} and ε_{zz} are equal to 0 too.

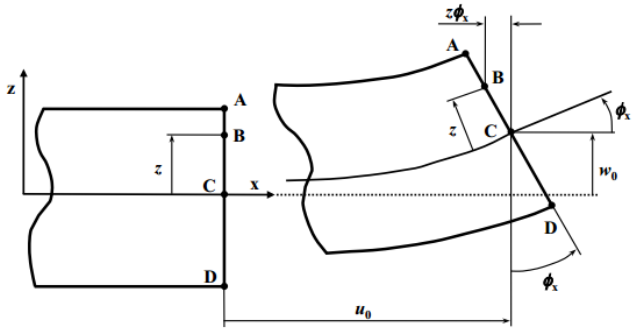


Figure 5 Mid plane deformation

3.6.1. Laminate deformation

Every deformation in a certain distance z of mid plane, can be calculated if you know mid plane deformation $[\varepsilon^0_{xyz}]$ and curvature $[\kappa_{xyz}]$, as it's expressed in Equation 10.

$$\left[\varepsilon_{xyz}\right]=\left[\varepsilon^{0}_{xyz}\right]+z\left[\kappa_{xyz}\right]$$

$$\begin{bmatrix}\varepsilon_{xx}\\ \varepsilon_{yy}\\ \gamma_{xy}\end{bmatrix}=\begin{bmatrix}\varepsilon^{0}_{xx}\\ \varepsilon^{0}_{yy}\\ \gamma^{0}_{xy}\end{bmatrix}+z\begin{bmatrix}\kappa_{xx}\\ \kappa_{yy}\\ \kappa_{xy}\end{bmatrix} \quad \text{(Eq.10)}$$

Equation 10 Deformation calculus

A deformation characteristic from composite materials is that they're constants throughout z, as it's showed in Figure 6.

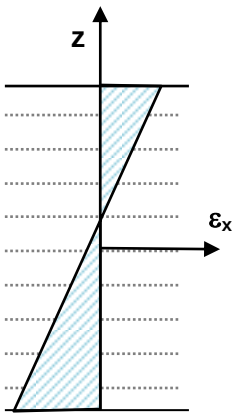


Figure 6 Graphical representation of ε_{xx} and z

3.6.2. Laminate stresses

We can know every point of the laminate's stress because we have the deformation and there is an equation which represents stress and middle plane deformation and curvature:

$$\left[\sigma\right]^{k}=\left[\overline{Q}\right]^{k}\left[\varepsilon^{0}\right]+z\left[\overline{Q}\right]^{k}\left[\kappa\right]$$

Equation 11 Stress calculus knowing deformations (Eq.11)

Stress representation is not continuous because of the material properties (including orientation) concern on stress calculation. We have an example in Figure 7.

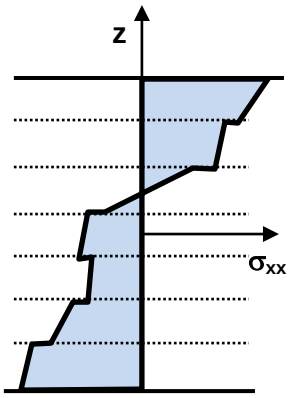


Figure 7 Graphical representation of stress σ_{xx} throughout z

3.7. Laminate constitutive equation

We can calculate the mid plane strains and curvatures using the laminate constitutive equation (Equation 12) knowing normal forces (N) and torque (M).

$$\begin{bmatrix} N \\ M \end{bmatrix} = \begin{bmatrix} A & B \\ B & D \end{bmatrix} \begin{bmatrix} \varepsilon^0 \\ \kappa \end{bmatrix} \qquad \text{(Eq.12)}$$

Equation 12 Laminate constitutive equation

This matrix is obtained by using the A, B and D matrix (explained next) creating a matrix that multiplied by the strains and curvatures will give us the normal forces and torques. We can also use this equation as we explained before (to calculate these strains and curvatures). The extended equation is showed at Equation 13.

$$\begin{bmatrix} N_x \\ N_y \\ N_{xy} \\ \hline M_x \\ M_y \\ M_{xy} \end{bmatrix} = \left[\begin{array}{ccc|ccc} A_{11} & A_{12} & A_{13} & B_{11} & B_{12} & B_{13} \\ A_{21} & A_{22} & A_{23} & B_{21} & B_{22} & B_{23} \\ A_{31} & A_{32} & A_{33} & B_{31} & B_{32} & B_{33} \\ \hline B_{11} & B_{12} & B_{13} & D_{11} & D_{12} & D_{13} \\ B_{21} & B_{22} & B_{23} & D_{21} & D_{22} & D_{23} \\ B_{31} & B_{32} & B_{33} & D_{31} & D_{32} & D_{33} \end{array}\right] \begin{bmatrix} \varepsilon_x^0 \\ \varepsilon_y^0 \\ \gamma_{xy}^0 \\ \hline \kappa_x \\ \kappa_y \\ \kappa_{xy} \end{bmatrix} \qquad \text{(Eq.13)}$$

Equation 13 Laminate constitutive equation extended

3.7.1. A, B and D matrixes

A, B and D matrixes are really important because by using them we can calculate the mechanical response of the laminate, like mid plane deformations and curvatures. Every matrix represents a different material property.

- A matrix: Deformation plane's stiffness' matrix
- B matrix: Compliance matrix
- D matrix: Bending stiffness matrix

A matrix can be calculated as the sum of every layer's stiffness multiplied by its thickness. (Equation 14)

$$A_{ij} = \sum_{k=1}^{N} \overline{Q}_{ij}^{k} \left(z_k - z_{k-1} \right) \qquad \text{(Eq.14)}$$

Equation 14 A matrix calculus

B matrix can be calculated as the sum of every layer's stiffness multiplied by the difference of every squared z (the top squared z minus the bot squared z) and divided by two. (Equation 15)

$$B_{ij} = \frac{1}{2} \sum_{k=1}^{N} \overline{Q}_{ij}^{k} \left(z_k^2 - z_{k-1}^2 \right) \qquad \text{(Eq.15)}$$

Equation 15 B matrix calculus

$$D_{ij} = \frac{1}{3} \sum_{k=1}^{N} \overline{Q}_{ij}^{k} \left(z_k^3 - z_{k-1}^3 \right) \qquad \text{(Eq.16)}$$

Equation 16 D matrix calculus

4. POLAR PLOT OF THE ENGINEERING CONSTANTS

One of the laminate's properties is that its stiffness changes depending on the force and direction. The best way to illustrate the dependence of the elastic constants on the laminate orientation is using the polar plot. Using polar plot bring us a global vision about stiffness and other properties as we can see at Figure 8.

We need to know Young modulus on the xx axis (E_{xx}), Young modulus in the yy axis (E_{yy}) in this case (our case) $E_{yy} = E_{xx}$ so we'll only represent one of them, shear modulus (G_{xy}) and Poisson coefficient (ν_{xy}).

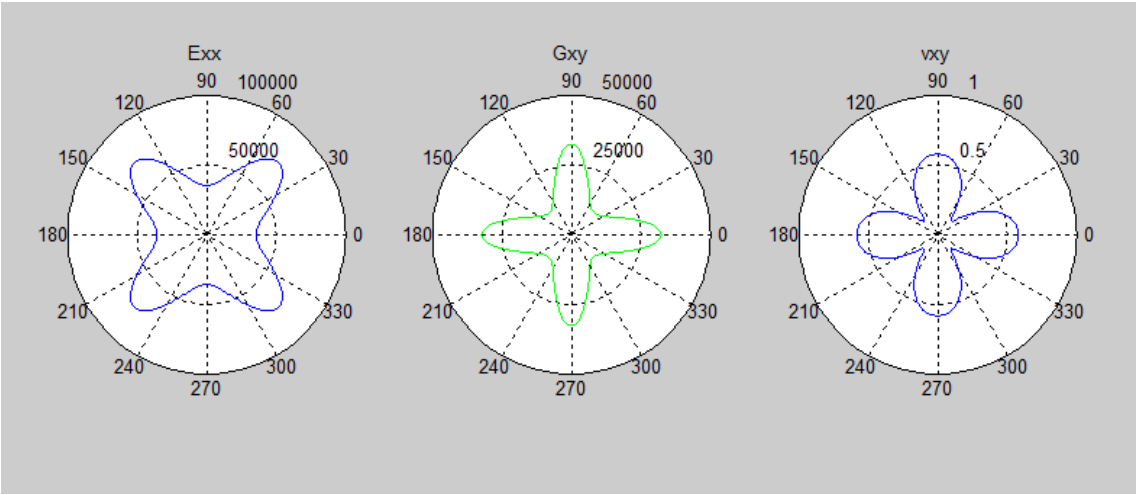


Figure 8 Variable's polar plot.

When we know these properties we'll represent them on the radius and the orientation on our θ variable (rotation).

4.1. Plain elastic properties

We can calculate all the in-plane properties that we need to represent on polar plots from the elements of the A matrix and the total thickness (*2h*)

$$E_{xx} = \frac{1}{2hA_{11}} \quad \text{(Eq.17)}$$

Equation 17 E_{xx} calculus

$$G_{xy} = \frac{1}{2hA_{33}} \quad \text{(Eq.18)}$$

Equation 18 G_{xy} calculus

$$\nu_{xx} = -\frac{A_{12}}{A_{22}} \quad \text{(Eq.19)}$$

Equation 19 ν_{xy} calculus

4.2. Bending elastic properties

We can calculate easily bending elastic properties when knowing D matrix and again total thickness (*2h*).

$$E_{xxf} = \frac{12}{(2h)^3 D_{11}} \quad \text{(Eq.20)}$$

Equation 20 Bending E_{xx} calculus

$$G_{xyf} = \frac{12}{(2h)^3 D_{33}} \quad \text{(Eq.21)}$$

Equation 21 Bending G_{xy} calculus

$$\nu_{xyf} = -\frac{D_{12}}{D_{22}} \quad \text{(Eq.22)}$$

Equation 22 Bending ν_{xy} calculus

5. BUCKLING ANALYSIS

The most important property for a skin's fuselage is the critical buckling load. In order to obtain quick results calculating the critical buckling load we'll use the closed-form from Rayleigh-Ritz energy method[2]. We prefer this alternative instead of the expensive computational structural analysis because we'll calculate it lots of times when running the optimization algorithm. So the critical load can be calculated using the Equation 23.

$$N_x^{cr} = K_x \frac{\pi^2}{b_p^2} \sqrt{D_{11} D_{22}} \qquad \text{(Eq.23)}$$

Equation 23 Critical buckling load calculus

Where b_p is the plate width and K_x is a non-dimensional buckling coefficient dependent on the elements of the bending stiffness matrix:

$$K_x = 2\sqrt{1 - 4\delta\gamma - 3\delta^4 + 2\delta^2\beta} + 2\left(\beta - 3\delta^2\right) \qquad \text{(Eq.24)}$$

Equation 24 Definition of buckling coefficient K_x

Where:

$$\beta = \frac{D_{21} + 2D_{33}}{\sqrt{D_{11} D_{22}}} \qquad \text{(Eq.25)}$$

Equation 25 β definition

$$\delta = \frac{D_{23}}{\sqrt[4]{D_{11} D_{22}^3}} \qquad \text{(Eq.26)}$$

Equation 26 δ definition

$$\gamma = \frac{D_{13}}{\sqrt[4]{D_{11}^3 D_{22}}} \qquad \text{(Eq.27)}$$

Equation 27 γ definition

6. ANT COLONY OPTIMIZATION (ACO)

In this chapter we'll explain how the algorithm works. This is the algorithm that we'll use to calculate the different stacking sequences.

In order to verify our code, we'll reproduce the problem that it's solved by N. Kogiso[3] and we realised that we obtained some similar results. So we know our algorithm is working properly.

6.1. ACO concept

Our objective is to find a laminate which can assume the demanding stresses having similar properties to the baseline but reducing its thickness. In that way we could reduce its weight. The critical buckling load must be equal or higher than the baseline. The way to explore the different possibilities and to choose the best one is using the ant colony optimization (ACO) adapted to our need[4]. This algorithm studies and reproduces the ant behaviour when they seek for food.

Ants choose a random path in order to find food. When they return to their colony, they leave a pheromone trail. This pheromone trail, the shorter the path is, the stronger the pheromone smell is. It helps to the next expedition to follow the best path. But they don't only go to these paths, they also try new paths, but the strongest it's the path, the most probable is that ants follow it. So at least, the most of the ants will follow the shortest path to the food.

6.2. How do ants choose a path?

As we mentioned before, initially ants choose the path randomly. But the probability to choose a path or another is conditioned by the already existing pheromone smell on the path (τ_{ij}). When the ant is in a node, it looks for every possibility to go to the next node but always moving forward. This process is repeated till they arrive to the last node. So the mathematic way to represent this fact is showed in Equation 28.

$$p_{ij}^{(k)} = \frac{\tau_{ij}}{\sum \tau_{ij}} \qquad \text{(Eq.28)}$$

Equation 28 Choosing path probability calculus

Where $p_{ij}^{(k)}$ is the probability of the chosen path, τ_{ij} is the pheromone trail for a certain variable.

When the ant arrives to the last node, it returns to the colony on the same path leaving the pheromone trail $\Delta\tau^{(k)}$. All of these pieces of path (between nodes) will be updated as it's showed in Equation 29.

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau^{(k)} \quad \text{(Eq.29)}$$

Equation 29 Updating path's pheromone trail

6.2.1. Pheromone evaporation

The chosen paths are random (especially first chosen paths) and as a consequence these solutions may not be the best ones. So, we must explore new paths in order to assure that we will choose the best one. Ants can do it because when an ant goes through a path, the pheromone trail is evaporated in a certain percentage (*p*), so this path becomes less interesting for the next ant. Our solution will be, again, copying this ant's behaviour in order to explore different solutions.

6.2.2. Path quality definition

As it was explained before, ants have more probabilities to follow a certain path if the pheromone in the trail is higher than in the other possible paths. The amount of pheromone increases depending on the path quality (the shorter, the better) and the number of ants that are walking through this path. That's because every ant leaves a quantity of pheromone. When a path is the best (or really close to it), eventually most of the ants of the colony will go through this path. The mathematic representation of this pheromone update is the Equation 30

$$\tau_{ij} = (1-\rho)\tau_{ij} + \sum_{k=1}^{N} \Delta\tau_{ij}^{(k)} \quad \text{(Eq.30)}$$

Equation 30 Updating pheromone trail of a path

$$\Delta\tau_{ij}^{(k)} = \frac{Q}{L_k} \quad \text{(Eq.31)}$$

Equation 31 Pheromone increase calculus

Q is a constant and L_k is the length of the path.

The shortest paths (the best solutions) have a more intense pheromone smell, so the most of the colony will choose these paths.

6.3. ACO applied to the optimization of the stacking sequence of composite laminates

- **1st step:** First of all we should choose the parameter values. We have to set how many ants (*N)* will search for a solution in every sub-iteration, the number of complete iterations (*I*) the pheromone trail evaporation (ρ) and the importance of the results (ζ). And last but not least we have to set the initial pheromone matrix τ_{ij}. Once we have done this, we can initialize our iteration variable (*I=1*).
- **2nd step:** Choosing the path probability depends on the pheromone trail. For every node we'll have all our variables parametrized with a number between 0 and 1. So we'll have the initial variable of every layer between 0 and a certain value, the last variable between a value (smaller than 1) and 1. So we'll create a range of values, necessary in step 3 to choose every layer's orientation.

$$p_{ij} = \frac{\tau_{ij}^{(l)}}{\sum_{m=1}^{p} \tau_{im}^{(l)}}; i = 1,2,...,n; j = 1,2,...,p \qquad \text{(Eq.32)}$$

Equation 32 Choosing path probability calculus

Where *n* is the number of variables (possible orientations) and *p* the number of layers.

- **3rd step:** In order to choose a path, for every ant, we'll create random numbers. We'll create as random numbers as numbers of layers have our laminate. These random numbers are between 0 and 1. As probability choosing path parameter is also values between 0 and 1, we'll choose the orientation which has this number in its probability range.
- **4th step:** When every of our ants (N) have already explored a solution, we'll calculate their objective function and save the best one (function value and sequence found) and the worst (only the function value).
- **5th step:** If the best solution isn't explored by the most of our ants (I choose a minimum of *N/100*) we'll repeat the 4 steps before (sub-iteration) and we'll update the pheromone matrix (Equation 33). In order to update this matrix, we have to know the evaporated pheromone ($\tau_{ij}^{(old)}$), which is calculated in Equation 34, and the increase of the best pheromone trail (Equation 35). We will just increase the amount of pheromone in the best solution. In the other hand, if it's explored by the most of the ants, we'll consider our function good enough. And we'll record our optimum results and we'll start again from step 1 reinitializing our best function, the best and worst function values and the pheromone matrix. This proceed will be repeated *I* times (as many as the number of iterations we fixed before).

$$\tau_{ij}^{(l)} = \tau_{ij}^{(old)} + \sum_{k} \Delta\tau_{ij}^{(k)} \quad \text{(Eq.33)}$$

Equation 33 Pheromone matrix update

$$\tau_{ij}^{(old)} = (1-\rho)\tau_{ij}^{(l-1)} \quad \text{(Eq.34)}$$

Equation 34 Evaporation calculus

$$\Delta\tau_{ij} = \frac{\zeta f_{best}}{f_{worst}} \quad \text{(Eq.35)}$$

Equation 35 Increase of pheromone trail calculus

6.4. Algorithm improvements

In order to adapt the algorithm to our necessities, I have done the following modifications.

6.4.1. Choosing path conditions

When we randomly generate a laminate, it has to follow two rules:

- Ply clustering is not allowed. It means that it is not allowed that a more than a quarter of the total number of layers have the same orientation.
- We will explore three different kind of laminates:
    - Completely asymmetrical
    - Completely asymmetrical with 8% rule[1]
    - Symmetrical

In step 3, when we choose a path, if one of these conditions is not accomplished, our algorithm will assign a new random number for the layer which is breaking the rules. If this layer is still breaking the rules, the algorithm will assign a different random number till the layer is satisfying the rules.

6.4.2. Objectives and constraints

Our objective function will be to reduce to 0 the B matrix. But we also need that the critical buckling load satisfies a certain minimum value. Another necessity is that stiffness has to be similar to the baseline.

If one of these conditions isn't accomplished, we'll apply a penalty to our objective function. When the buckling load doesn't achieve the value that the baseline has (it's the minimum) we'll apply a penalization of 10^3. When the stiffness is not similar to the baseline's stiffness, we apply a penalization of 10^2. For the stiffness constraint we have a 50% of tolerance between the values, if the new value is higher or lower than this 50%, we apply the penalization.

The objective function is defined in Equation 36. As we can see, if B matrix is completely 0, the objective function will be 1. So if any penalization is applied, we can see which constraint is broken.

$$f(x) = \sum B_{ij}^2 + 1 \quad where\, i, j \in [1,3] \tag{Eq.36}$$

Equation 36 Objective function definition

[1] The 8% rule says that the laminate have to have a minimum of an 8% of the total layers for every possible orientation. Normally it's used when we use the conventional orientations of 0º, ±45º, 90º.

6.4.3. Pheromone matrix update

As we said before, our objective function is to minimize matrix B to 0. So when we will update our pheromone matrix, it doesn't make sense to do it as we showed in Equation 35, because the path will be better when f_{best} is close to 0. In order to fix this fact, we'll calculate the increase of pheromone as we show in Equation 37.

$$\Delta\tau_{ij} = \frac{f_{worst}}{\zeta f_{best}} \qquad \text{(Eq.37)}$$

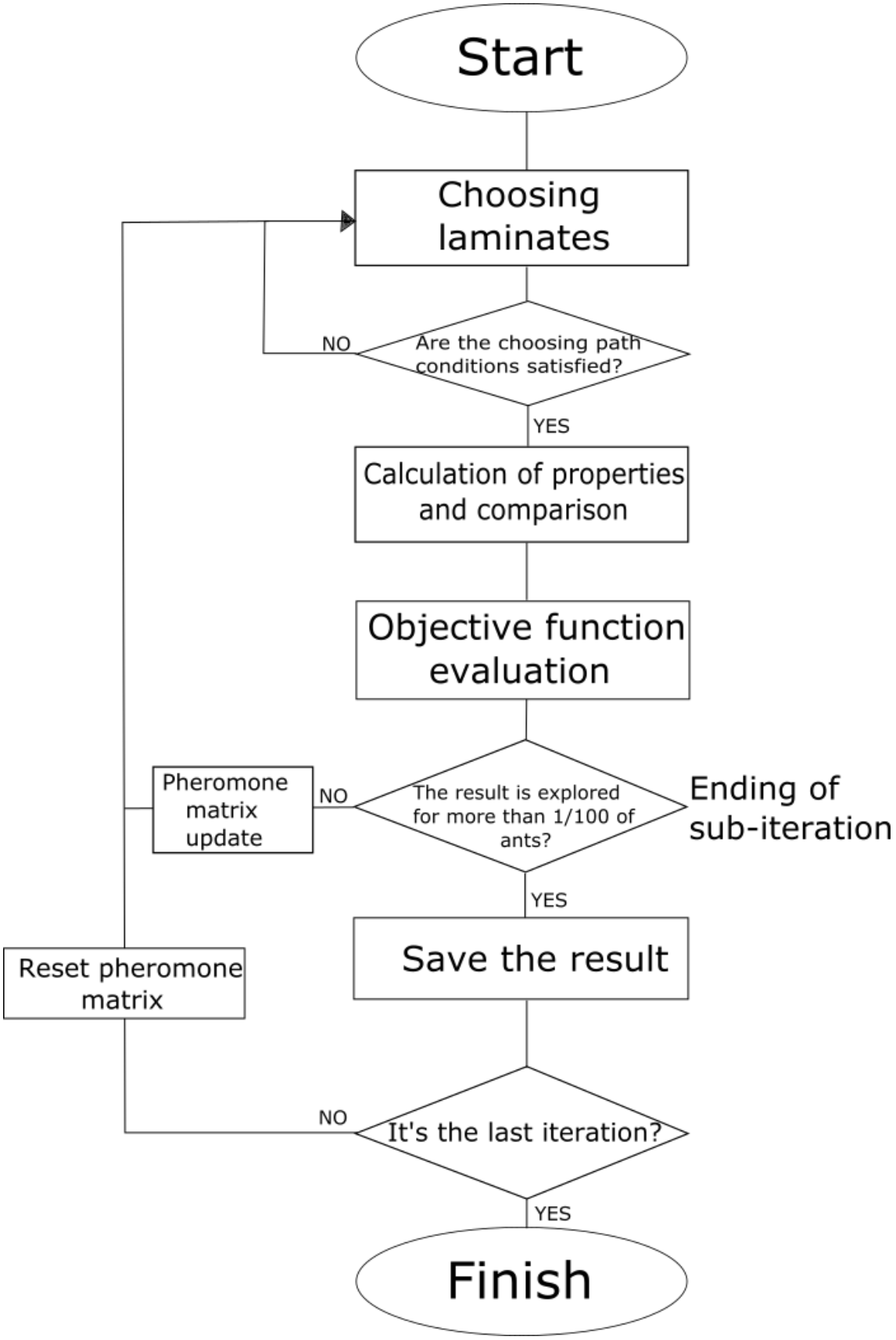
Equation 37 Increase of pheromone adapted

6.5. Algorithm's flow chart

Before starting with the chart's steps, we have to set the next variables:

- Nº of ants
- Nº of iterations
- Baseline
- Evaporation (ρ)
- Results importance (ζ)

The flow chart is showed on next page.


Start
Choosing laminates
NO
Are the choosing path conditions satisfied?
YES
Calculation of properties and comparison
Objective function evaluation
Pheromone matrix update
NO
The result is explored for more than 1/100 of ants?
Ending of sub-iteration
YES
Save the result
Reset pheromone matrix
NO
It's the last iteration?
YES
Finish

7. PROBLEM DESCRIPTION

After the introduction of the theoretical background, we're going to define the studied practical problem. The aim of this project is to improve the skin of aeroplane's fuselages by reducing its weight. Fuselages are built with the skin and with stiffener elements. As these elements have been build lots of times, they have found some laminates that works really well. These laminates are called baselines. We're going to explore alternative layouts breaking the established rules in order to find a thinner laminate with better properties than the basic. These rules that we're going to break are for example the 8% rule (explained before), the symmetrical layouts (we will also explore this possibility, but we will explore different ways) and the balanced laminates (for every layer orientation must be another layer with its opposite, for example 45º and -45º).

7.1. Defining the fuselage

In the fuselage there are two kinds of stiffeners: a longitudinal one (stringer) and a circumferential one (frame). In a composite fuselage, stringer and skin are made of composite materials fastened by metal straps. The fuselage is divided by panels with two or more stiffeners connected by the skin, as it's showed at Figure 9.

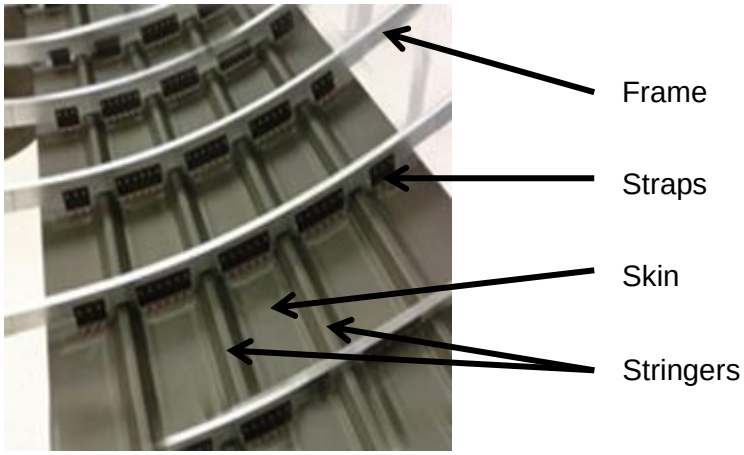


Figure 9 Example of aircraft fuselage (from San Diego Composites' web)

The way to design our laminate is to calculate the skin between straps and stringers. If you know the properties necessary of every panel of skin, you can calculate one of them and use it for the rest of the fuselage's skin. The skin's parameter will be defined in Figure 10:

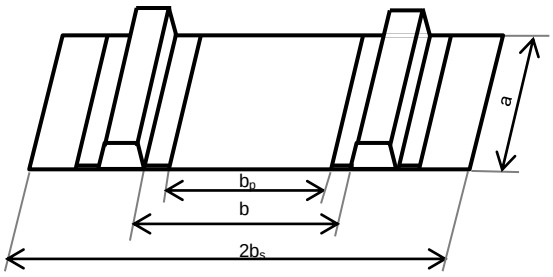


Figure 10 Skin panel's geometry

In our case the geometry of the panel is:

- Stringer pitch (b_p) = 200 mm
- Stringer foot (S_f) = 28.6 mm
- Skin width (b) = 134.3 mm
- Panel length (a) = 635 mm

7.2. Defining the laminate

The materials used is IMA/M21E material. We'll use this material because they're using thin plies technology and we'll use it to minimize the laminate thickness. This material presents an excellent damage tolerance. It's composed of epoxy reinforced with high tensile strength reinforced carbon fibres.

7.2.1. Material

The material used will be the IMA/M21E. The properties are the following:

E_{11}	$154 \cdot 10^3$
E_{22}	$8{,}50 \cdot 10^3$
G_{12}	$4{,}20 \cdot 10^3$
μ_{12}	$0{,}35$

Table 1 Material's properties in [MPa]

7.2.2. Thin plies

Normally, the thinnest conventional ply thickness is 0'12 [mm], but with the new technology, it appears these new kinds of plies: thin plies. These plies are really interesting because it can reduce the thickness in a 1/6 of the conventional minimum thickness. That means that with the same laminate thickness (so the same weight) we can have more layers.

Using these thin plies we have lots of possibilities to create new laminates that can be thinner than the usually used and with the same properties. We will use them in the following way. We will set a number of plies bigger than the baseline's one, and if we improves the mechanical specifications, we will modify the laminate thickness in order to reduce weight (care should be taken because the mechanical properties are worse when the thickness is lower).

In our case we will find laminates with 20 plies because when we explore the asymmetrical ones, it's possible to reduce[2] the B matrix to 0.

7.2.3. Baseline stacking sequence

There are two stacking sequences that are the most used in these cases. Both of these layouts are chosen to improve different properties. One has its compression buckling (CB) optimized (following some rules) and the other is optimized for the damage tolerance (DT). Both are laminates formed by 13 plies of 0.127 mm (ply thickness) so they have a total thickness of 1.651 mm. The ply orientations are the classical ones: (0º, 90º, ±45º) and as it's said before, using the rules of compliances (symmetrical, balanced and using the 8% rule). The stacking sequences are the following:

a)
45
-45
-45
45
90
0
90
0
90
45
-45
-45
45

b)
45
-45
90
-45
0
45
90
45
0
-45
90
-45
45

Figure 11 Baseline stacking sequences a) laminate optimized for compression buckling b) laminate optimized for the damage tolerance

[2] This is the only number of layers that we could reduce the B matrix to 0. This number had been found empiricaly.

There are different rules that were followed in order to build these laminates. There are three plies that are the same in both laminates: a 45º ply in bottom and top and a 90º just in the middle of the laminate (layer 7). There's also a very restrictive rule, they must have a specific percentage of every ply orientation. As we can see at Figure 11, in both laminate are:

- 4 layers of 45º orientation
- 4 layers of -45º orientation
- 3 layers of 90º orientation
- 2 layers of 0º orientation

They are represented in different colours to make it more visual.

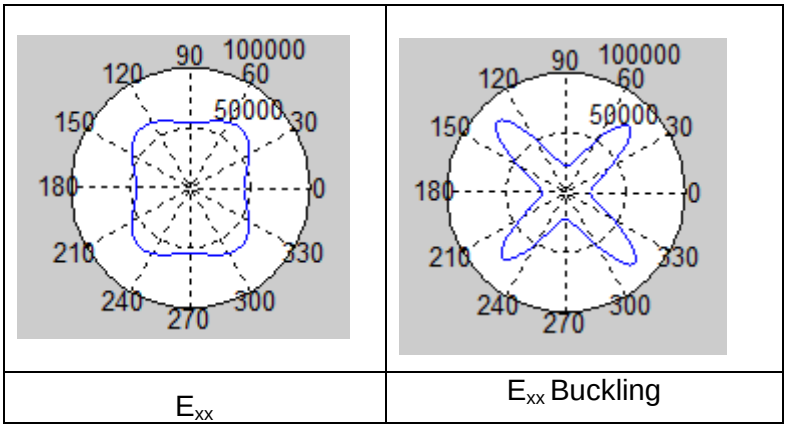
They have another restrictive peculiarity: the laminate must be symmetric and balanced in order to avoid warpage when creating the laminate and having orthotropic characteristics. As we have explained, it's really important to have all the B_{ij}=0 in our laminate. It's because if it's not zero, when building the laminate, it can appear torsion and traction deformations (warpage).

But before checking our case, we modified the optimization algorithm in order to demonstrate that we can have the same properties with laminates that are completely anti-symmetrical. So we'll have freedom to explore the different possibilities of laminates and found the best. In fact we'll focus on anti-symmetrical laminates because symmetrical and balanced laminates had been studied many times.

7.2.4. Baseline properties

Firstly we will analyse the CB stacking sequence:

- sequence = [45,-45,-45,45,90,0,90,0,90,45,-45,-45,45]
- Layer thickness = 0.127 [mm]
- Total thickness = 1.651 [mm]
- N_x^{cr} (critical buckling load) = 29,48 [N/mm]
- Stiffness

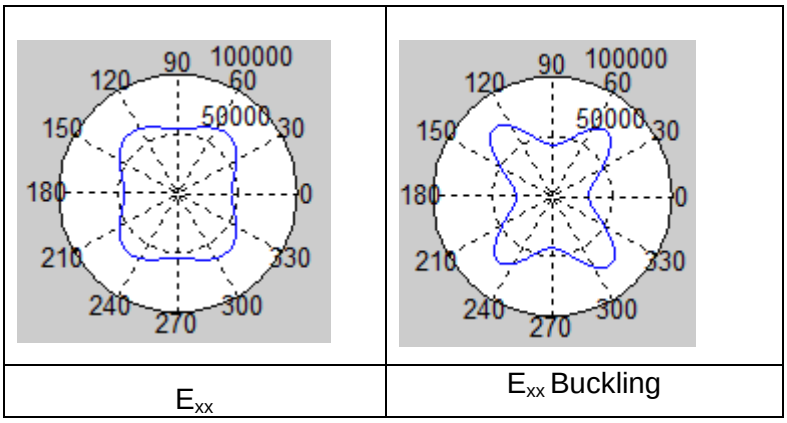


E_{xx} | E_{xx} Buckling

Figure 12 Elastic properties of the baseline laminate optimized for buckling (CB)

And here we will analyse the DT stacking sequence:

- sequence = [45,-45,90,-45,0,45,90,45,0,-45,90,-45,45]
- Layer thickness = 0.127 [mm]
- Total thickness = 1.651 [mm]
- N_x^{cr} (critical buckling load) = 27,22 [N/mm]
- Stiffness [MPa]



E_{xx} | E_{xx} Buckling

Figure 13 Elastic properties of the baseline laminate optimized for damage tolerance

As we're trying to maximize our critical buckling load, we'll compare our results with the CB baseline.

8. RESULTS

8.1. Introduction

Finally, some results had been found. In order to find if it's possible to improve the critical buckling load, we will compare all our results with the CB baseline. That's because of the CB buckling is the one that they use when they need a high critical buckling load.

We explored different kind of laminates:

- Symmetrical laminates
- Totally asymmetrical laminates
- Totally asymmetrical laminates with a minimum of an 8% of layers for every orientation

First of all we will analyse the laminate one by one and finally we'll compare all the results in a table.

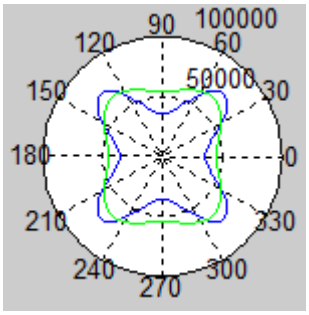
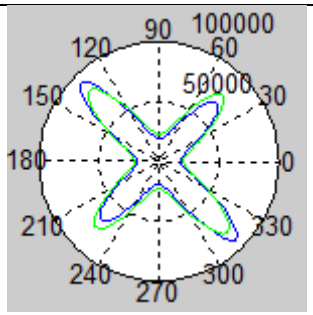
All the results that we've selected as possible result have B_{ij}=0.

We'll compare the mechanical properties in a polar diagram where the green function corresponds to the baseline's properties.

8.2. Symmetrical results

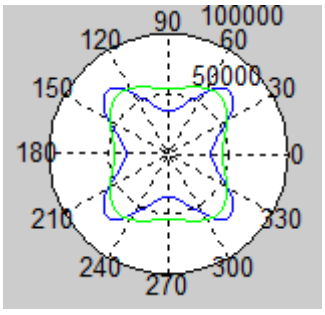
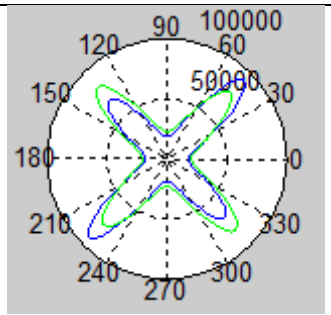
This chapter aims to explore whether some symmetrical laminates exist that can improve the critical buckling load:

- Seq. = [45,-45,45,-45,45,-45,45,90,0,-45,-45,0,90,45,-45,45,-45,45,-45,45]
- Layer thickness = 0.082 [mm]
- Total thickness = 1.651 [mm]
- N_x^{cr} (critical buckling load) = 29,58 [N/mm]
- Stiffness [MPa]

	
E_{xx}	E_{xx} Buckling

Seq. = [-45,45,-45,45,-45,45,-45,90,0,45,45,0,90,-45,45,-45,45,-45,45,-45]

- Layer thickness = 0.082 [mm]
- Total thickness = 1.651 [mm]
- N_x^{cr} (critical buckling load) = 29,58 [N]
- Stiffness [MPa]

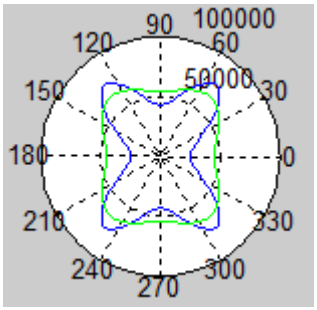
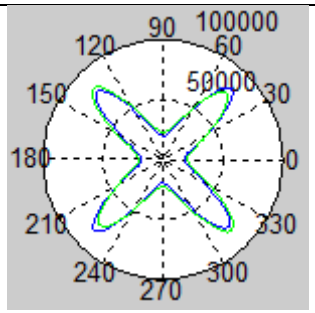
	
E_{xx}	E_{xx} Buckling

As we can see, we can improve a bit the critical buckling load (29,58 N/mm in front of 29,48 N/mm for the CB baseline).

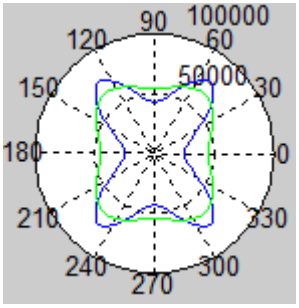
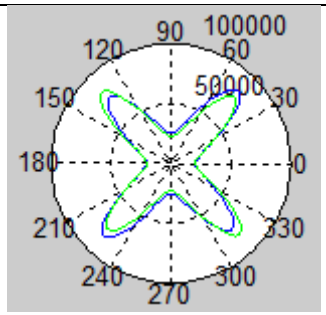
8.3. Totally asymmetrical layout

Normally laminates are symmetrical and balanced in order to make 0 the B matrix. But as we said, we have explored different laminates which are asymmetrical and achieve the B=0. That was a really important discover. It increases a lot the different possibilities of making laminates. We can see different asymmetrical solutions with B_{ij}=0.

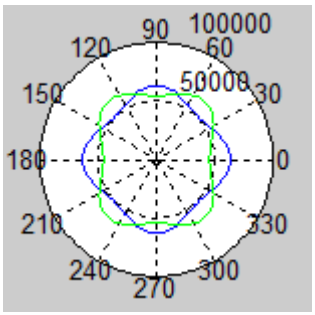
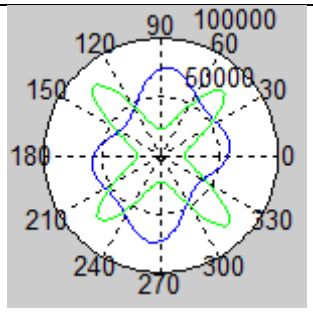
- Sequence = [45,-45,-45,45,-45,45,90,-45,45,90,45,90,90,-45,-45,45,-45,45,45,-45]
- Layer thickness = 0.082 [mm]
- Total thickness = 1.651 [mm]
- N_x^{cr} (critical buckling load) = 29,84 [N]
- Stiffness [MPa]

	
E_{xx}	E_{xx} Buckling

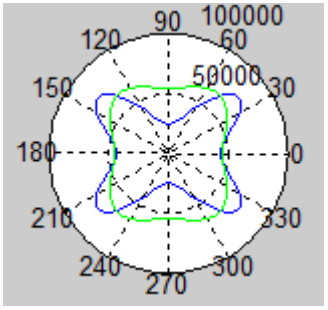
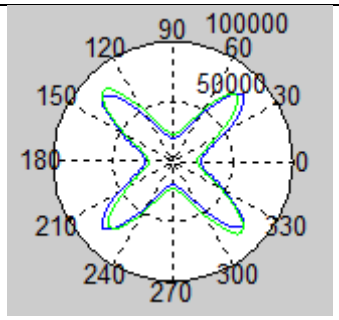
- Seq. = [-45,-45,45,45,-45,45,90,90,45,45,90,-45,45,-45,-45,90,-45,-45,45,45]
- Layer thickness = 0.082 [mm]
- Total thickness = 1.651 [mm]
- N_x^{cr} (critical buckling load) = 29,2 [N]
- Stiffness [MPa]

	
E_{xx}	E_{xx} Buckling

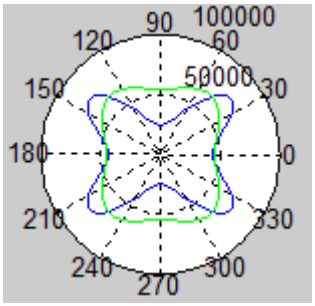
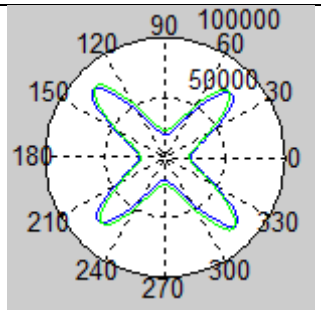
- sequence = [90,0,90,-45,0,-45,45,45,45,0,90,-45,90,0,0,90,0,45,90,-45]
- Layer thickness = 0.082 [mm]
- Total thickness = 1.651 [mm]
- N_x^{cr} (critical buckling load) = 21,49[N]
- Stiffness [MPa]

	
E_{xx}	E_{xx} Buckling

- Seq. = [-45,45,45,-45,-45,0,45,45,-45,0,45,0,-45,0,-45,45,45,-45,-45,45]
- Layer thickness = 0.082 [mm]
- Total thickness = 1.651 [mm]
- N_x^{cr} (critical buckling load) = 29,55 [N]
- Stiffness [MPa]

	
E_{xx}	E_{xx} Buckling

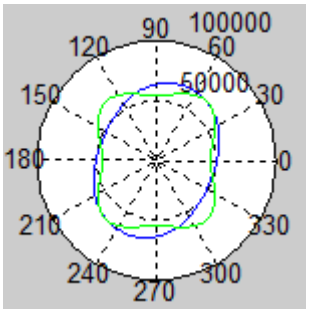
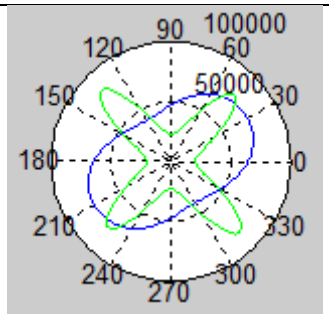
- sequence = [45,-45,-45,45,-45,45,45,0,0,0,-45,-45,45,-45,0,45,-45,45,45,-45]
- Layer thickness = 0.082 [mm]
- Total thickness = 1.651 [mm]
- N_x^{cr} (critical buckling load) = 29,66[N]
- Stiffness [MPa]

	
E_{xx}	E_{xx} Buckling

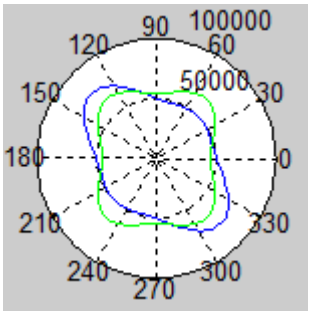
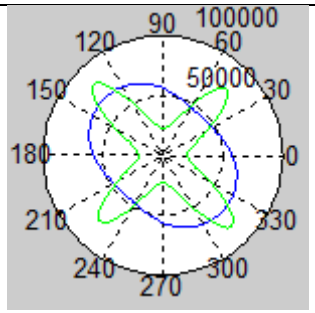
As we can see some of the results have really different E_{xx} if we compare it with the baseline. So we will see the obtained results with the "8% rule".

8.4. Totally asymmetrical layout with 8% rule

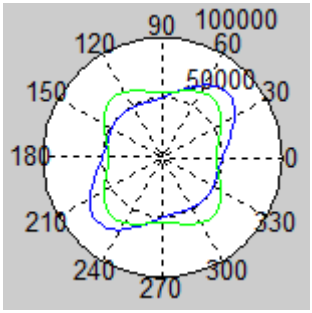
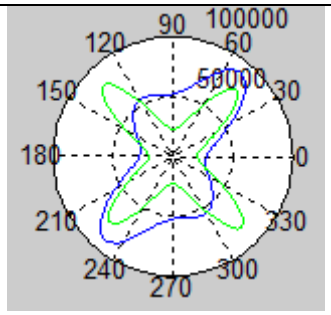
- Sequence = [0,-45,-45,45,90,0,90,45,90,90,-45,-45,90,45,-45,45,0,0,90,-45]
- Layer thickness = 0.082 [mm]
- Total thickness = 1.651 [mm]
- N_x^{cr} (critical buckling load) = 22,77 [N]
- Stiffness [MPa]

	
E_{xx}	E_{xx} Buckling

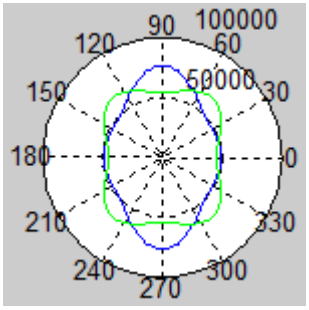
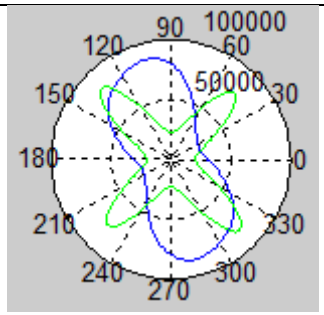
- Sequence = [90,45,0,-45,45,0,-45,45,45,90,45,90,0,45,-45,-45,45,45,90,0]
- Layer thickness = 0.082 [mm]
- Total thickness = 1.651 [mm]
- N_x^{cr} (critical buckling load) = 23,17[N]
- Stiffness [MPa]

	
E_{xx}	E_{xx} Buckling

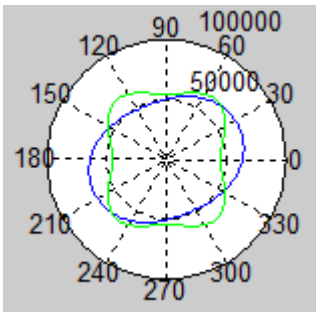
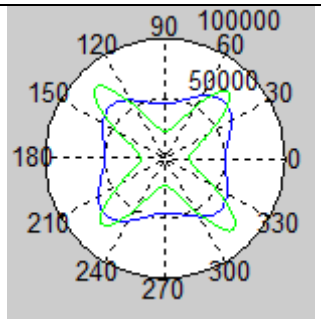
- Sequence = [-45,90,-45,45,45,-45,0,-45,90,0,90,0,0,-45,45,-45,-45,45,-45,90]
- Layer thickness = 0.082 [mm]
- Total thickness = 1.651 [mm]
- N_x^{cr} (critical buckling load) = 25,17[N]
- Stiffness [MPa]

	
E_{xx}	E_{xx} Buckling

- Sequence = [45,90,90,45,-45,0,90,0,90,-45,90,0,-45,-45,90,0,90,45,45,90]
- Layer thickness = 0.082 [mm]
- Total thickness = 1.651 [mm]
- N_x^{cr} (critical buckling load) = 20,09[N]
- Stiffness [MPa]

	
E_{xx}	E_{xx} Buckling

- Sequence = [0,45,-45,-45,45,90,90,-45,0,90,0,-45,0,0,0,-45,45,45,90,-45]
- Layer thickness = 0.082 [mm]
- Total thickness = 1.651 [mm]
- N_x^{cr} (critical buckling load) = 25,6[N]
- Stiffness [MPa]

	
E_{xx}	E_{xx} Buckling

As we can see here, some of the results have similar E_{xx} but, in contrast of the other results, we couldn't improve the critical buckling load.

8.5. Global comparison

Here we will compare all our results with the baseline in order to illustrate the improvement (or not) in the critical load for the optimized laminate:

ASYMMETRICAL SOLUTIONS:		
Sequence	**N_{x_crit}**	**$N_x/N_{x\_baseline}$**
[45,-45,-45,45,-45,45,90,-45,45,90,45,90,90,-45,-45,45,-45,45,45,-45]	29,84	1,012
[45,-45,-45,45,-45,45,45,0,0,0,-45,-45,45,-45,0,45,-45,45,45,-45]	29,66	1,006
[-45,45,45,-45,-45,0,45,45,-45,0,45,0,-45,0,-45,45,45,-45,-45,45]	29,55	1,002
[-45,-45,45,45,-45,45,90,90,45,45,90,-45,45,-45,-45,90,-45,-45,45,45]	29,2	0,990
[90,0,90,-45,0,-45,45,45,45,0,90,-45,90,0,0,90,0,45,90,-45]	21,49	0,728

ASYMMETRICAL SOLUTIONS WITH 8% RULE:		
Sequence	**N_{x_crit}**	**$N_x/N_{x\_baseline}$**
[0,45,-45,-45,45,90,90,-45,0,90,0,-45,0,0,0,-45,45,45,90,-45]	25,6	0,868
[-45,90,-45,45,45,-45,0,-45,90,0,90,0,0,-45,45,-45,-45,45,-45,90]	25,17	0,853
[90,45,0,-45,45,0,-45,45,45,90,45,90,0,45,-45,-45,45,45,90,0]	23,17	0,785
[0,-45,-45,45,90,0,90,45,90,90,-45,-45,90,45,-45,45,0,0,90,-45]	22,77	0,772
[45,90,90,45,-45,0,90,0,90,-45,90,0,-45,-45,90,0,90,45,45,90]	20,09	0,681

SYMMETRICAL SOLUTIONS:		
Sequence	**N_{x_crit}**	**$N_x/N_{x\_baseline}$**
[45,-45,45,-45,45,-45,45,90,0,-45,-45,0,90,45,-45,45,-45,45,-45,45]	29,58	1,003
[-45,45,-45,45,-45,45,-45,90,0,45,45,0,90,-45,45,-45,45,-45,45,-45]	29,58	1,003

As we can see at "$N_x/N_{x\_baseline}$" column we're improving just with symmetrical solutions and asymmetrical solutions. When we apply the 8% rule with asymmetrical, we can't improve the N_x.

9. BUDGET SUMMARY

The execution total cost is **FIVE THOUSAND THREE HUNDRED AND FIVETY FOUR EUROS AND SEVENTY THREE CENTS (5.354,73 €)**

Girona, 25 de Maig de 2015

10. CONCLUSION AND FUTURE WORK

As we could see on the results, we can improve the critical buckling load, but not enough if we want to reduce the laminate thickness. So as a conclusion, our results show us that the baseline is really good and we can improve it just a bit:

- When we explored the asymmetrical solutions without following any rule, we could improve just a 1'2% the critical buckling load.
- When we explored the asymmetrical solutions but following the 8% rule, we couldn't improve the critical buckling load, but the other mechanical properties were better than the laminates explored when just exploring asymmetrical cases. In this case we could just achieve a buckling load that can support just the 86'8% of the baseline's critical buckling load.
- Finally we explored the symmetrical cases. We could just reach the same critical buckling load as the baseline.

As we can see, increasing the layout from 13 layers to 20 doesn't give us an important improvement. It can be also that the way to calculate the critical buckling load is too simplified. So as a future work we could compute the extended formula to calculate the buckling load. In order to do this we will need more powerful computers. Another way to trying to improve our results is to break more rules, for example trying to use atypical angles (15º, 30º, etc.) and see how can we improve our layout but maintaining the B_{ij}=0.

11. BIBLIOGRAPHY

[1] HERAKOVIC,Carl T. Mechanics of fibrous composites. Wiley & Sons (1998)
[2] F. AMERICH, M. SERRA. Optimization of laminate stacking sequence for maximum buckling load using the ant colony optimization (ACO) metaheuristic. Composites: Part A 39 (2008) 262–272
[3] N. KOGISO, L.T. WATSON, Z. GÜRDAL, R.T. HAFTKA. Genetic algorithms with local improvement for composite laminate design. Structural Optimization 7, 207-218 Springer Verlag (1994)
[4] F. AMERICH, M. SERRA. An ant colony optimization algorithm for stacking sequence design of composite laminates. CMES – Comput Model Eng Sci 2006;13(1):49–66

12. GLOSSARY

σ = Stresses

ε = Strain

κ = Curvature

C, Q = Material stiffness matrix

S = Compliance matrix

T = Transformation matrix

N = Applied forces

M = Applied torque

A = Plane deformation stiffness matrix

B = Compliance matrix

D = Bending stiffness matrix

E = Young modulus

G = Shear modulus

ν = Poisson coefficient

h = Half of the laminate thickness

N_x^{cr} = Critical buckling load

K_x = Non dimensional buckling coefficient

τ = Pheromone trail

p = Choosing path probability

ρ = Evaporation parameter

ζ = Result importance parameter

b_p = Stringer pitch

S_f = Stringer foot

b = Skin width

a = Panel length

n = number of layers

z = layer's position vector

ANNEX A: CALCULUS

A1. A Matrix

```
% Compute A matrix
% Ultima revisio 28 Octubre

function A = matA(n,Qb,z)
A = zeros(3,3);
for i=1:n;
    for j=1:3;
        for k=1:3;
            A(j,k) = A(j,k) + Qb(j,k,i)*(z(i+1) - z(i));
        end;
    end;
end;
end
```

A2. B Matrix

```
% Compute B matrix
% Ultima revisio 28 Octubre

function B = matB(n,Qb,z)
B = zeros(3,3);
for i=1:n;
    for j=1:3;
        for k=1:3;
            B(j,k) = B(j,k) + Qb(j,k,i)*(z(i+1)^2 - z(i)^2)*(1/2);
        end;
    end;
end;
end
```

A3. D Matrix

```
% Compute D matrix
% Ultima revisio 28 Octubre

function D = matD(n,Qb,z)
D = zeros(3,3);
for i=1:n;
    for j=1:3;
        for k=1:3;
            D(j,k) = D(j,k) + Qb(j,k,i)*(z(i+1)^3 - z(i)^3)*(1/3);
        end
    end
end
end
```

A4. ABBD Matrix

```
%revisat 29 octubre
function ABBD=matABBD(A,B,D)
    for i=1:3
        for j=1:3
            ABBD(i,j)=A(i,j);
        end
        for j=4:6
            ABBD(i,j)=B(i,j-3);
        end
    end
    for i=4:6
        for j=1:3
            ABBD(i,j)=B(i-3,j);
        end
        for j=4:6
            ABBD(i,j)=D(i-3,j-3);
        end
    end
end
```

A5. Strain calculus

```
%revisat 29 octubre
function epsilon=epsilon(matDef)
    for i=1:3
        epsilon(i,1)=matDef(i);
    end
end
```

A6. Curvature calculus

```
%revisat 29 octubre
function k=k(matDef)
    for i=1:3
        k(i,1)=matDef(i+3);
    end
end
```

A7. Mechanical properties

```
% Revisio 6 Novembre
% recalcul de la matriu A i D

function [Exx, Gxy, Vxy, Exxf, Gxyf, Vxyf, rot] =
Apolar(sq,n,S,z,theta,h)
a = 360/theta;
for i=0:a
    sq_new = sq + theta*i;
    T_new = matT(sq_new, n);
    Tgamma_new=matTgamma(T_new,n);
    Qb_new=matQb(T_new,S,n,Tgamma_new);
    A_new = matA(n,Qb_new,z);
    a_new = inv(A_new);
    D_new = matD(n,Qb_new,z);
    d_new = inv(D_new);

    Exx(i+1)=1/(a_new(1,1)*h);
    Gxy(i+1)=1/(a_new(3,3)*h);
    Vxy(i+1)=-a_new(1,2)/a_new(2,2);

    Exxf(i+1)=12/(d_new(1,1)*(h)^3);
    Gxyf(i+1)=12/(d_new(3,3)*(h)^3);
    Vxyf(i+1)=-d_new(1,2)/d_new(2,2);

    rot(i+1) = theta*i*pi/180;
end
end
```

A8. Deformation matrix for every layer

```
% Ultima modificacio 27 Octubre

function matDef_capa=matDef_capa(matDef,zrep)
    for i=1:length(zrep)
        matDef_capa(:,i)=matDef+zrep(i)*matDef;
    end
end
```

A9. K matrix

```
function k=matk(matDef)
    for i=1:3
        k(i)=matDef(i+3);
    end
end
```

A10. Z matrix

```
% Laminate bounds
% Location of the bounds of the laminae and laminate's midplane

function z = matZ(sqt)
z = 0;
suma = 0;
for i=1:length(sqt)
    suma = sqt(i) + suma;
end
mig = suma/2;
for i=1:length(sqt)
    z(i,1) = mig - suma;
    suma = suma - sqt(i);
end
z(length(sqt)+1,1)=mig;
end
```

A11. Z bot, mid, top calculus

```
%Calcul de Z per bot mid top.
%Revisio 31 Octubre
function zbmt=zbmt(z,sqt,n)
zbmt(1)=z(1);
zbmt(2)=z(1)+0.5*sqt(1);
zbmt(3)=z(1)+sqt(1);
for i=1:n-1
    zbmt(i*3+1)=z(i+1);
    zbmt(i*3+2)=z(i+1)+0.5*sqt(i+1);
    zbmt(i*3+3)=z(i+1)+sqt(i+1);
end
end
```

A12. S Matrix

```
% Compute compliance matrix
% Ultima revisio 28 Octubre

function S = matS(E11,E22,nu12,G12)
S=[1/E11 -nu12/E11 0;
   -nu12/E11 1/E22 0;
   0 0 1/G12];
end
```

A13. T Matrix

```
% Compute transformation matrices T
% Ultima revisio 28 Octubre

function T = matT(sq,n)
for k=1:n
    theta = sq(k)*pi/180;
    T(:,:,k)=[ cos(theta)^2  sin(theta)^2  2*cos(theta)*sin(theta);
               sin(theta)^2  cos(theta)^2 -2*cos(theta)*sin(theta);
               -cos(theta)*sin(theta)  cos(theta)*sin(theta)
cos(theta)^2-sin(theta)^2];
end
end
```

A14. T_γ Matrix

```
% Compute transformation matrices Tgamma
% Ultima revisio 28 Octubre

function Tgamma = matTgamma(T,n)
for k=1:n;
    Tgamma(:,:,k)= (inv(T(:,:,k)))';
end
end
```

A15. Q_b Matrix

```
% Compute stiffness transformed matrices
% Ultima revisio 28 Octubre

function Qb = matQb(T,S,n,Tgamma)
for k=1:n
    Sb(:,:,k)=inv(Tgamma(:,:,k))*S*T(:,:,k);
    Qb(:,:,k)=inv(Sb(:,:,k));
end
end
```

A16. Transformed Q_b matrix

```
function Qbt=matQbt(Qb,k)
for i=1:3
    for j=1:3
        Qbt(i,j,k)=Qb(i,j);
    end
end
end
```

A17. Global stresses for every layer

```
%revisat 4 Novembre
function sigma_global_capa=sigma_global_capa(n,epsilon,kurvature,Qb,z)
    for a=1:n
        sigma_global_capa(:,a)=Qb(:,:,a)*epsilon +
z(a)*Qb(:,:,a)*kurvature;
    end
```

A18. Local σ for every layer

```
%revisat 4 Novembre
function sigma_local_capa=sigma_local_capa(n,epsilon,Q)
    for a=1:n
        sigma_local_capa(:,a)=Q(:,:)*epsilon(:,a);
    end
```

ANNEX B: ACO FUNCTIONS

B1. Random numbers

```
%numeros random
%revisat 20 Novembre
%comprovat

function r=r(n_form,n_layers)
clear r
r=zeros(n_layers,n_form);
for i=1:n_layers
    for j=1:n_form
        r(i,j)=rand;
    end
end
```

B2. Calculation of probabilities

```
%calcul de probabilitat Pij
%revisio 20 Novembre
%comprovat

function [p_ini,p_fin] = calcul_p(n_layers, n_var, mat_feromona)
p_ini=zeros(n_layers,n_var);
p_fin=zeros(n_layers,n_var);
    for i=1:n_layers
        sum_tau=0;
        for j=1:n_var
            sum_tau=sum_tau+mat_feromona(i,j);
        end
        for j=1:n_var
            if j==1
                  p_ini(i,j)=0;
                  p_fin(i,j)=mat_feromona(i,j)/sum_tau;
            else
                  p_ini(i,j)=p_fin(i,j-1);
                  p_fin(i,j)=p_fin(i,j-1)+(mat_feromona(i,j)/sum_tau);
            end
        end
    end
end
```

B3. Choosing a path (Symmetrical case)

```
% triar cami
% Revisio 21 Novembre

function mat_cami=mat_cami_sym(n_layers,n_var,n_form,r,p_ini,p_fin,n)
mat_cami=zeros(n_layers,n_var,n_form);
sum=zeros(n_var,n_form);
for k=1:n_form
    ply_clus=0;
    for i=1:n_layers/2
        ok = 0;
        cont = i-n_layers/2;
        i_sym = ((n_layers/2)+1)-cont;
        while ok == 0
```

```
            for j=1:n_var
                if r(i,k)>p_ini(i,j) && r(i,k)<p_fin(i,j)
                    if i>1
                        if mat_cami(i-1,j,k)==1
                            ply_clus = ply_clus+1;
                        else
                            ply_clus = 0;
                        end
                    end
                    if ply_clus > n/6
                       r(i,k) = rand;
                    else
                        mat_cami(i,j,k)=1;
                        mat_cami(i_sym,j,k)=1;
                        ok = 1;
                    end
                else
                    mat_cami(i,j,k)=0;
                end
            end

        end

    for n=1:n_var
        sum(n,k)=(sum(n,k)+mat_cami(i,n,k))*2;
    end
    end
    while ok ~= 2
    for n=1:n_var
            if sum(n,k)<0.08*n_layers*2
                for j=1:n_var
                    if sum(j,k)>0.16*n_layers
                        for i=1:n_layers
                            if mat_cami(i,j,k)==1
                               cont = i-n_layers/2;
                               i_sym = ((n_layers/2)+1)-cont;
                               mat_cami(i,j,k)=0;
                               mat_cami(i_sym,j,k)=0;
                               mat_cami(i,n,k)=1;
                               mat_cami(i_sym,n,k)=1;
                               break
                            end
                        end
                    end
                end
            end
    end
    if n==n_var
       ok = 2;
    end
    end


end
end
```

B4. Choosing a path (Asymmetrical case)

```
% triar cami
% Revisio 21 Novembre

function mat_cami=mat_cami_asym(n_layers,n_var,n_form,r,p_ini,p_fin,n)
mat_cami=zeros(n_layers,n_var,n_form);
sum=zeros(n_var,n_form);
for k=1:n_form
    ply_clus=0;
    for i=1:n_layers
        ok = 0;
        cont = i-n_layers/2;
        i_sym = ((n_layers/2)+1)-cont;
        while ok == 0
            for j=1:n_var
                if r(i,k)>p_ini(i,j) && r(i,k)<p_fin(i,j)
                    if i>1
                        if mat_cami(i-1,j,k)==1
                            ply_clus = ply_clus+1;
                        else
                            ply_clus = 0;
                        end
                    end
                    if ply_clus > n/4
                       r(i,k) = rand;
                    else if i<(n_layers/2+0.5)
                        mat_cami(i,j,k)=1;
                        ok = 1;
                        else if i>n_layers/2 && mat_cami(i_sym,j,k)==0
                                mat_cami(i,j,k)=1;
                                ok = 1;
                            else
                                r(i,k) = rand;
                            end
                        end
                    end
                else
                    mat_cami(i,j,k)=0;
                end
            end

        end
    end


end
end
```

B5. Choosing a path (8% asymmetrical case)

```
% triar cami
% Revisio 21 Novembre

function
mat_cami=mat_cami_8perc(n_layers,n_var,n_form,r,p_ini,p_fin,n)
mat_cami=zeros(n_layers,n_var,n_form);
sum=zeros(n_var,n_form);
for k=1:n_form
    ply_clus=0;
    for i=1:n_layers
        ok = 0;
        cont = i-n_layers/2;
        i_sym = ((n_layers/2)+1)-cont;
        while ok == 0
            for j=1:n_var
                if r(i,k)>p_ini(i,j) && r(i,k)<p_fin(i,j)
                    if i>1
                        if mat_cami(i-1,j,k)==1
                            ply_clus = ply_clus+1;
                        else
                            ply_clus = 0;
                        end
                    end
                    if ply_clus > n/4
                       r(i,k) = rand;
                    else if i<(n_layers/2+0.5)
                        mat_cami(i,j,k)=1;
                        ok = 1;
                        else if i>n_layers/2 && mat_cami(i_sym,j,k)==0
                                mat_cami(i,j,k)=1;
                                ok = 1;
                            else
                                r(i,k) = rand;
                            end
                        end
                    end
                else
                    mat_cami(i,j,k)=0;
                end
            end
             for n=1:n_var
                sum(n,k)=sum(n,k)+mat_cami(i,n,k);
             end
        end
    end
    while ok ~= 2
    for n=1:n_var
            if sum(n,k)<0.08*n_layers
                for j=1:n_var
                    if sum(j,k)>0.16*n_layers
                        for i=1:n_layers
                            if mat_cami(i,j,k)==1
                               mat_cami(i,j,k)=0;
                               mat_cami(i,n,k)=1;
                               break
                            end
                        end
                    end
                end
```

```
            end
    end
    if n==n_var
       ok = 2;
    end
    end

end
end
```

B6. N_x^{cr} Calculus (variable func)

```
% Objective function calculus
% Critical buckling load

function [func,K_x] = func_k(D11,D22,D33,D12,D13,D23,bp)
      beta = (D12+2*D33)/(sqrt(D11*D22));
      delta = D23/((D11*D22^3)^(1/4));
      gamma = D13/((D11^3*D22)^(1/4));
      K_x = 2*sqrt(1-4*delta*gamma-
      (3*delta^4)+2*delta^2*beta)+2*(beta-3*delta^2);
      func=K_x*((3.1415^2)/(bp^2))*sqrt(D11*D22);
```

B7. Mechanical properties comparison

```
% Comparacio de rigidesa a traccio i a flexio

function [similar,Exx,Exxf,Exx_ref,Exxf_ref] =
comparison_E_Ef(sq,sq_ref,n,n_ref,S,z,z_ref,h,h_ref,tol_prop_elast,the
ta)
a = 360/theta;
tolerance = tol_prop_elast/100;
similar = 1;
for i=0:a
    sq_new = sq + theta*i;
    sq_ref_new = sq_ref + theta*i;

    % sq
    T_new = matT(sq_new, n);
    Tgamma_new=matTgamma(T_new,n);
    Qb_new=matQb(T_new,S,n,Tgamma_new);
    A_new = matA(n,Qb_new,z);
    D_new = matD(n,Qb_new,z);

    % sq_ref
    T_new_ref = matT(sq_ref_new, n_ref);
    Tgamma_new_ref=matTgamma(T_new_ref,n_ref);
    Qb_new_ref=matQb(T_new_ref,S,n_ref,Tgamma_new_ref);
    A_new_ref = matA(n_ref,Qb_new_ref,z_ref);
    D_new_ref = matD(n_ref,Qb_new_ref,z_ref);

    Exx(i+1)=1/(A_new(1,1)*h);
    Exx_ref(i+1)=1/(A_new_ref(1,1)*h_ref);

    Exxf(i+1)=12/(D_new(1,1)*(h)^3);
    Exxf_ref(i+1)=12/(D_new_ref(1,1)*(h_ref)^3);
     if Exx > Exx_ref*tolerance
         similar = 0;
     else if Exx < Exx_ref*tolerance
             similar = 0;
         end
```

```
     end
     if Exxf > Exxf_ref*tolerance
         similar = 0;
     else if Exxf < Exxf_ref*tolerance
             similar = 0;
         end
     end

end
end
```

B8. Function evaluation

```
% Evaluacio funcio i tria f_best, f_worst
% Revisat 24 Novembre

function [f_best,f_worst,sq_best,rep, mat_cami_best, Exx_best,
Exxf_best,Carrega_critica_best]=eval_f2_fuselage(sq,f_best,f_worst,rep
,sq_best,k,f,mat_cami,mat_cami_best,it,n,Exx,Exxf,Exx_best,Exxf_best,C
arrega_critica,Carrega_critica_best)
    % Evaluacio de funcio
    if sq(k,:)==sq_best(it,:)
    rep=rep+1;
    end
    if f==f_best(it)
        if Carrega_critica>Carrega_critica_best(it)
            f_best(it)=f;
            sq_best(it,:)=sq(k,:);
            mat_cami_best(:,:)=mat_cami(:,:,k);
            Exx_best(it,:)=Exx(1,:);
            Exxf_best(it,:)=Exxf(1,:);
            Carrega_critica_best(it)=Carrega_critica;
        end
    end
    if f<f_best(it)
        f_best(it)=f;
        sq_best(it,:)=sq(k,:);
        mat_cami_best(:,:)=mat_cami(:,:,k);
        Exx_best(it,:)=Exx(1,:);
        Exxf_best(it,:)=Exxf(1,:);
        Carrega_critica_best(it)=Carrega_critica;
    else if f>f_worst
        f_worst=f;
        mat_cami_best=mat_cami_best;
        Exx_best(it,:)=Exx_best(it,:);
        Exxf_best(it,:)=Exxf_best(it,:);
        Carrega_critica_best(it)=Carrega_critica_best(it);
        end
    end
end
```

B9. Pheromone evaporation

```
% Evaporate pheromone
% Ultima revisio 17 Novembre
% comprovat

function mat_fer_old = ev_pheromone(mat_feromona,rho)
mat_fer_old=mat_feromona*(1-rho);
end
```

B10. Pheromone matrix update

```
% Actualitzacio de la matriu feromona
% Revisio 24 Novembre

function
      [increment,mat_feromona]=mat_fer_min_abs(sigma,f_best,f_worst,ma
      t_fer_old,mat_cami_best,rep,it)
      delta_tau=sigma*abs(1/(f_best(it)/f_worst));
      increment=delta_tau*(1+rep)*mat_cami_best;
      mat_feromona=mat_fer_old+increment;
end
```

ANNEX C: VERIFYING CODES

C1. ACO Verifcation

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%%%%%            Autor:           Axel           Garcia           Garcia
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% TFDG: Optimitzacio d'orientacio de capa
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Ultima revisio 16 Desembre
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Revisat: G.Guillamet, 15 December
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%

clear all;
format short e

%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%%% DADES D'ENTRADA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%

%%%%% (S.I in mm)
%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
% % MATERIAL PROPERTIES
% % Graphite/epoxy
% E11     = 127.59e3;
% E22     =  13.03e3;
% nu12    =  0.30;
% G12     =  6.41e3;
% eps11u  =  0.008/1.5;
% eps22u  =  0.029/1.5;
% gam12u  =  0.015/1.5;
% t_lay = 0.127;              % Ply thickness [mm]
%
% % BIAXIAL COMPRESSION PARAMETERS
% a  = 508.;          % Plate length (x-direction)
% b  = 127.;          % Plate width (y-direction)
% mm = 7;             % Number of possible waves in the length
direction
% nn = 7;             % Number of possible waves in the width direction
% Nx = 175.e-3;       % Distributed load in the x-ditection
% Ny = 21.875e-3;     % Distributed load in the y-ditection

%%%%% (Imperial units)
%%%%%%%%%%%%%%%%%%%%%
% MATERIAL PROPERTIES
% Graphite/epoxy
E11     = 18.5e6;
E22     =  1.89e6;
nu12    =  0.30;
G12     =  0.93e6;
eps11u  =  0.008/1.5;
```

```
eps22u  =  0.029/1.5;
gam12u  =  0.015/1.5;
t_lay = 0.005;     % Ply thickness [in]
n_layers = 12;     % A quarter of number of plies because it's
symmetric
                   % and we do packs of 2 plies

% BIAXIAL COMPRESSION PARAMETERS
a  = 20.;      % Plate length (x-direction)
b  = 5.;       % Plate width (y-direction)
mm = 7;        % Number of possible waves in the length direction
nn = 7;        % Number of possible waves in the width direction
Nx = 1.;       % Distributed load in the x-ditection
Ny = Nx/8.;    % Distributed load in the y-ditection

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Ant Colony Optimization %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Finding the best laminate working on a biaxial case %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Initializing variables of ACO
n_form = 100;  % Number of ants
n_it = 10;     % Number of iterations
finish = 0;
f_best = zeros(1,n_it);
f_worst = 0;
rep = 0;
f_optima = 0;
sq_best = zeros(n_it,n_layers*4);

% Variable matrix for ACO calculus
vars = [0 45 90]; % For every +30 or +45 we have an inmediate -30 or -
45
                     % and for every 0 or 90 we have an identic
inmediately
for i = 1:n_layers
    mat_var(i,:) = vars;
end

n_var = length(vars);
mat_cami_best = zeros(n_layers,n_var);
mat_feromona = ones(n_layers,n_var);
sqt = ones(n_layers*4,1)*t_lay;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%%% Aplication of ACO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
for it=1:n_it
    rep = 0;
    finish = 0;
    mat_cami_best = zeros(n_layers,n_var);
    mat_feromona = ones(n_layers,n_var);
    f_worst = 0;

    while finish==0
        %% Random number generator
```

```
clear r
r=r(n_form,n_layers);

%% Calculculus of probability choosing a path
[p_ini,p_fin]=calcul_p(n_layers,n_var,mat_feromona);

%% Choosing a path
clear mat_cami
mat_cami=mat_cami(n_layers,n_var,n_form,r,p_ini,p_fin);

%% Transforming the path into a laminate
for k=1:n_form
    for i=1:n_layers
        for j=1:n_var
            if mat_cami(i,j,k)==1
                if j==1 || j==n_var
                    sq(k,i*2-1) = mat_var(i,j);
                    sq(k,i*2) = mat_var(i,j);
                else
                    sq(k,i*2-1) = mat_var(i,j);
                    sq(k,i*2) = -mat_var(i,j);
                end
            end
        end
    end
end

% Simmetry of laminate
for k=1:n_form
    for i=1:24
        sq(k,24+i) = sq(k,25-i);
    end
end

for k=1:n_form
    %% Calculs de S, Q, T, Qb, Sb, A, B, D i ABBD.
    n=n_layers*4;
    z=matZ(sqt);

    % Stiffness and compliance matrices (material CSYS)
    S = matS(E11,E22,nu12,G12);
    Q = inv(S);

    % Transformation matrices
    T = matT(sq(k,:),n);
    Tgamma = matTgamma(T,n);

    % Transformed stifness and compliance matrices
    Qb = matQb(T,S,n,Tgamma);

    % Matrix D
    D = matD(n,Qb,z);
    D11 = D(1,1);
    D22 = D(2,2);
    D33 = D(3,3);
    D12 = D(1,2);

    % Matrix a
    A = matA(n,Qb,z);
```

```
            mat_a = inv(A);
            a11 = mat_a(1,1);
            a22 = mat_a(2,2);
            a33 = mat_a(3,3);
            a12 = mat_a(1,2);

            %% Lambda calculus

            % Buckling lambda
            clear lambda_cb
            lambda_cb = lambda_cb(D11,D12,D33,D22,Nx,Ny,a,b,mm,nn);

            % Failure criterion
            clear lambda_cf
            lambda_cf = lambda_cf(a11, a12, a22, sq, eps11u, eps22u);

            % Choosing the worst case
            lambda_crit(k) = min(lambda_cb,lambda_cf);
        end
        %% Evaluation of objective function
        [f_best,f_worst,sq_best,rep,
mat_cami_best]=eval_f(sq,f_best,f_worst,rep,sq_best,n_form,lambda_crit
,mat_cami,mat_cami_best,it);

        %% Pheromone evaporation
        % Choosing a rho
        rho=0.1;
        mat_fer_old=ev_pheromone(mat_feromona,rho);
        %% Pheromone matrix actualization
        % Choosing a sigma
        sigma=0.7;

mat_feromona=mat_fer(sigma,f_best,f_worst,mat_fer_old,mat_cami_best,re
p,it);
        if rep==200
            finish=1;
        end
            end
        end


%% Printing solution on command Window
sq_best          % Best laminate
f_best           % Lambda crit from best laminate
```

C2. N_x^{cr} Check

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%%%%% Autor: Axel Garcia Garcia
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% TFDG: Fuselage's skin optimization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%

clear all;
format short e

%% MATERIAL PROPERTIES
% IMA/M21E
E11     =  154.e3;      %[MPa]
E22     =  8.50e3;     %[MPa]
G12     =  4.2e3;     %[MPa]
nu12    =  0.35;

%% COMPRESSION PARAMETERS
bp = 200;           % Stringer width [mm]

%% Laminate parameters
sq = [45,-45,90,-45,0,45,90,45,0,-45,90,-45,45];
total_thickness = 1.651;
n = length(sq);
t_lay = total_thickness/n;
sqt = ones(n,1)*t_lay;
h = total_thickness;

%% Calculs de S, Q, T, Qb, Sb, A, B, D i ABBD.
% Stiffness and compliance matrices (material CSYS)
z = matZ(sqt);
S = matS(E11,E22,nu12,G12);
Q = inv(S);

% Transformation matrices
T = matT(sq,n);
Tgamma = matTgamma(T,n);

% Transformed stifness and compliance matrices
Qb = matQb(T,S,n,Tgamma);

% Matriu ABD
A = matA(n,Qb,z);
B = matB(n,Qb,z);
D = matD(n,Qb,z);
            D11 = D(1,1);
            D22 = D(2,2);
            D33 = D(3,3);
            D12 = D(1,2);
            D13 = D(1,3);
            D23 = D(2,3);
[carrega_critica,K_x] = func_k(D11,D22,D33,D12,D13,D23,bp)
```

ANNEX D: EXECUTABLE CODE (MAIN CODE)

D1. Calculation of the laminates for the fuselage

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%%%%% Autor: Axel Garcia Garcia
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% TFDG: Fuselage's skin optimization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%

clear all;
format short e

%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%%%%%% Material Properties
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%

%% MATERIAL PROPERTIES
% IMA/M21E
E11     = 154.00e3;      %[MPa]
E22     =  8.50e3;      %[MPa]
G12     =  4.20e3;       %[MPa]
nu12    =  0.35;


%% BIAXIAL COMPRESSION PARAMETERS
a  = 635;           % Plate length [mm] (x-direction)
b  = 134.3;         % Plate width [mm] (y-direction)
bp = 200;           % Stringer width [mm]


%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%%%% Ant Colony parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
rho = 0.1;
sigma=0.05;
n_form = 10000; % Number of ants
n_it = 10;     % Number of iterations
grau=45;      % Number of degree difference
sq_ref = [45,-45,-45,45,90,0,90,0,90,45,-45,-45,45];  % Lay-up de
referencia
n_ref = length (sq_ref);
tol_prop_elast = 50; % Enter in %(the tolerance will be +- the input
value)
theta = 45; % rotation angle for E and Ef comparison

%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%%%%% Ant Colony Optimization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Finding the best laminate for a fuselage's skin
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%

%% Initializing variables of ACO
finish = 0;
n_layers = 20;
total_thickness_ref = 1.651;            % [mm]
t_lay_ref = total_thickness_ref/n_ref;  % Ply thickness [mm]
t_lay = total_thickness_ref/n_layers;
f_best = ones(1,n_it)*(1e12);
f_worst = 1;
rep = 0;
nn_varr = 90/grau;
f_optima = 0;
sq_best = zeros(n_it,n_layers);
Exx_best=zeros(n_it,360/theta+1);
Exxf_best=zeros(n_it,360/theta+1);
Carrega_critica_best=zeros(n_it,1);

%%
% Variable matrix for ACO calculus
% vars(1) = 0;
% for i=1:nn_varr+1
%     vars(i+1) = vars(i)+grau;
% end
% vars(nn_varr+2)=-grau;
% for i=nn_varr+3:2*nn_varr
%     vars(i) = vars(i-1)-grau;
% end

vars = [0 45 90 -45];

for i = 1:n_layers
    mat_var(i,:) = vars;
end

n_var = length(vars);
mat_cami_best = zeros(n_layers,n_var);
sqt = ones(n_layers,1)*t_lay;
sqt_ref = ones(n_ref,1)*t_lay_ref;
sq = zeros(n_form,n_layers);
total_thickness = t_lay*n_layers;
h = total_thickness;
h_ref = total_thickness_ref;

%% Reference sequence calculus
n = n_layers;
z_ref=matZ(sqt_ref);

% Stiffness and compliance matrices (material CSYS)
S = matS(E11,E22,nu12,G12);
Q = inv(S);

% Transformation matrices
```

```
T_ref = matT(sq_ref,n_ref);
Tgamma_ref = matTgamma(T_ref,n_ref);

% Transformed stifness and compliance matrices
Qb_ref = matQb(T_ref,S,n_ref,Tgamma_ref);

% Matrix D
D_ref = matD(n_ref,Qb_ref,z_ref);
D11_ref = D_ref(1,1);
D22_ref = D_ref(2,2);
D33_ref = D_ref(3,3);
D12_ref = D_ref(1,2);
D13_ref = D_ref(1,3);
D23_ref = D_ref(2,3);
Carrega_critica_ref =
func_k(D11_ref,D22_ref,D33_ref,D12_ref,D13_ref,D23_ref,bp);

% Matrix a
A_ref = matA(n_ref,Qb_ref,z_ref);

% Matrix B
B_ref = matB(n_ref,Qb_ref,z_ref);

%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%%% Aplication of ACO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
for it=1:n_it
    rep = 0;
    finish = 0;
    mat_cami_best = zeros(n_layers,n_var);
    mat_feromona = ones(n_layers,n_var)*1000;
    f_worst = 1;
    disp(['Iteration: ', num2str(it)]);

    while finish==0
        %% Random number generator if sym, n_layers/2
        clear r
        r=r(n_form,n_layers/2);

        %% Calculculus of probability for choosing a path
        [p_ini,p_fin]=calcul_p(n_layers/2,n_var,mat_feromona);

        %% Choosing a path
        clear mat_cami
        mat_cami=mat_cami_sym(n_layers,n_var,n_form,r,p_ini,p_fin,n);


        %% Transforming the path into a laminate
        for k=1:n_form
            for i=1:n_layers
                for j=1:n_var
                    if mat_cami(i,j,k)==1
                        sq(k,i) = mat_var(i,j);
                    end
                end
```

```
            end
        end

        %% Calculus and evaluation of every ant sq properties

        for k=1:n_form
            %% Calculs de S, Q, T, Qb, Sb, A, B, D i ABBD.
            n = n_layers;
            z=matZ(sqt);

            % Stiffness and compliance matrices (material CSYS)
            S = matS(E11,E22,nu12,G12);
            Q = inv(S);

            % Transformation matrices
            T = matT(sq(k,:),n);
            Tgamma = matTgamma(T,n);

            % Transformed stifness and compliance matrices
            Qb = matQb(T,S,n,Tgamma);

            % Matrix D
            D = matD(n,Qb,z);
            D11 = D(1,1);
            D22 = D(2,2);
            D33 = D(3,3);
            D12 = D(1,2);
            D13 = D(1,3);
            D23 = D(2,3);

            % Matrix a
            A = matA(n,Qb,z);
            mat_a = inv(A);
            a11 = mat_a(1,1);
            a22 = mat_a(2,2);
            a33 = mat_a(3,3);
            a12 = mat_a(1,2);

            % Matrix B
            B = matB(n,Qb,z);
            B11 = B(1,1);
            B12 = B(1,2);
            B13 = B(1,3);
            B21 = B(2,1);
            B22 = B(2,2);
            B23 = B(2,3);
            B31 = B(3,1);
            B32 = B(3,2);
            B33 = B(3,3);

        %% Objective function calculus and comparison with a standard
laminate
            Carrega_critica = func_k(D11,D22,D33,D12,D13,D23,bp);
            func(k) =
B11^2+B22^2+B33^2+B12^2+B23^2+B13^2+B21^2+B32^2+B31^2+1;
            [similar,Exx,Exxf,Exx_ref,Exxf_ref] =
comparison_E_Ef(sq,sq_ref,n,n_ref,S,z,z_ref,h,h_ref,tol_prop_elast,the
ta);
        %% Calculus of restriction
        if Carrega_critica<Carrega_critica_ref
```

```
            func(k)=func(k)*1000;
        end
        if similar == 1
            f=func(k);
        else
            f=func(k)*100;
        end
        %% Evaluation of function
            [f_best,f_worst,sq_best,rep, mat_cami_best, Exx_best,
Exxf_best,Carrega_critica_best]=eval_f2_fuselage(sq,f_best,f_worst,rep
,sq_best,k,f,mat_cami,mat_cami_best,it,n,Exx,Exxf,Exx_best,Exxf_best,C
arrega_critica,Carrega_critica_best);
            mat_cami_best;
        end

        %% Pheromone evaporation
        % Choosing a rho
        mat_fer_old = ev_pheromone(mat_feromona,rho);
        %% Pheromone matrix actualization
        % Choosing a sigma

[increment,mat_feromona]=mat_fer_min_abs(sigma,f_best,f_worst,mat_fer_
old,mat_cami_best,rep,it);
        increment;
        mat_feromona;
        if rep>n_form/100
            finish=1;
            disp(['Optimum SQ: ']);
            disp(sq_best(it,:));
            disp(['Objective function value: ', num2str(f_best(it))]);
        end
    end
end

disp(['']);
disp(['Optimum results:']);
sq_best
f_best
```

D2. Code used in order to analyse results

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%%%%% Autor: Axel Garcia Garcia
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% TFDG: Optimitzacio d'orientacio de capa
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Ultima revisio 6 Novembre
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Revisat: G.Guillamet, 12 November
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%

clear all;
format short e


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%%% DADES D'ENTRADA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%

%% Propietats del material (lamina)(en MPa)
% IMA/M21E
E11     = 154.00e3;     %[MPa]
E22     =  8.503e3;      %[MPa]
G12     =  4.203e3;      %[MPa]
nu12    =  0.35;

%% Laminat: orientació i gruix
% Entrar sq en graus i gruix en mm
%sq = [45, -45, 45, -45, 90, 90, 45, -45, 45, -45, 45, -45, 0, 0, 45,
-45, 0, 0, 0, 0, 45, -45, 0, 0, 0, 0, 45, -45, 0, 0, 0, 0, 45, -45, 0,
0, 45, -45, 45, -45, 45, -45, 90, 90, 45, -45, 45, -45];
%for i=1:48
%    sqt(i) = 0.127;
%end
%sq=[0 45];
% sqt=[2 1];

% sq =  [-45    90    90     0    45   -45     0    45     0    45
45   -45    90     0   -45    90];

sq =[-45,45,-45,45,-45,45,-45,90,0,45,45,0,90,-45,45,-45,45,-45,45,-
45];  %baseline
t = 1.651/20;

sqt = ones(1,length(sq))*t;

h=0;
for i=1:length(sqt)
    h=h+sqt(i);
end
```

```
%% Càrregues (si entres N i M comentar epsilon i k, i viceversa)
Nx = 1;
Ny = 10;
Nxy = -20;

Mx = 60;
My = 20;
Mxy = 5;

NM = [Nx; Ny; Nxy; Mx; My; Mxy];

%epsilonxx=0;
%epsilonyy=0;
%gammaxy=0;
%kxx=0;
%kyy=0;
%kxy=0;
%matDef=[epsilonxx; epsilonyy; gammaxy; kxx; kyy; kxy;];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%%% Calculs
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
n = length(sq); % nombre de capes

% Coordenades de les interficies de cada capa
z = matZ(sqt);

% Coordenades en els punts d'integracio de cada capa
for i=1:n
    z_bot(i)=z(i);
end
for i=1:n
    z_mid(i)=z(i)+0.5*sqt(i);
end
for i=1:n
    z_top(i)=z(i+1);
end

% Coordenades dels punts d'integracio de cada capa (totes agrupades)
zbmt = zbmt(z,sqt,n);

%% Calculs de S, Q, T, Qb, Sb, A, B, D i ABBD.
% Stiffness and compliance matrices (material CSYS)
S = matS(E11,E22,nu12,G12);
Q = inv(S);

% Transformation matrices
T = matT(sq,n);
Tgamma = matTgamma(T,n);

% Transformed stifness and compliance matrices
Qb = matQb(T,S,n,Tgamma);

% Matriu ABD
A = matA(n,Qb,z);
B = matB(n,Qb,z)
```

```
D = matD(n,Qb,z);
            D11 = D(1,1);
            D22 = D(2,2);
            D33 = D(3,3);
            D12 = D(1,2);
            D13 = D(1,3);
            D23 = D(2,3);
ABBD = matABBD(A,B,D);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%%% Calculs
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% tensions i deformacions tenim els punts d'integració bot mid i top
%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%

%NM=ABBD*matDef;
matDef = inv(ABBD)*NM;
epsilon0 = epsilon(matDef); % Midplane deformations
kurvature = k(matDef);     % Curvatures

%%Càlcul de deformacions (CSYS GLOBAL x-y-z)
for i=1:length(zbmt)
    epsilon_bmt(:,i)=epsilon0 + zbmt(i)*kurvature;
end

%epsilon bot
epsilon_xyz_bot(:,1)=epsilon_bmt(:,1);
for i=1:n-1
    epsilon_xyz_bot(:,i+1)=epsilon_bmt(:,i*3+1);
end

%epsilon mid
epsilon_xyz_mid(:,1)=epsilon_bmt(:,2);
for i=1:n-1
    epsilon_xyz_mid(:,i+1)=epsilon_bmt(:,i*3+2);
end

%epsilon top
epsilon_xyz_top(:,1)=epsilon_bmt(:,3);
for i=1:n-1
    epsilon_xyz_top(:,i+1) = epsilon_bmt(:,i*3+3);
end

%%Calcul de deformacions (CSYS LOCALS 1-2-3)
for i=1:n
    epsilon_123_bot(:,i) = Tgamma(:,:,i)*epsilon_xyz_bot(:,i);
    epsilon_123_mid(:,i) = Tgamma(:,:,i)*epsilon_xyz_mid(:,i);
    epsilon_123_top(:,i) = Tgamma(:,:,i)*epsilon_xyz_top(:,i);
end

%%Càlcul de tensions (CSYS GLOBALS x-y-z)
sigma_xyz_bot = sigma_global_capa(n,epsilon0,kurvature,Qb,z_bot);
sigma_xyz_mid = sigma_global_capa(n,epsilon0,kurvature,Qb,z_mid);
sigma_xyz_top = sigma_global_capa(n,epsilon0,kurvature,Qb,z_top);
```

```
%% Càlcul de tensions (CSYS LOCALS 1-2-3)
sigma_123_bot = sigma_local_capa(n,epsilon_123_bot,Q);
sigma_123_mid = sigma_local_capa(n,epsilon_123_mid,Q);
sigma_123_top = sigma_local_capa(n,epsilon_123_top,Q);



%% Sigma xx bot mid i top juntes GLOBALS
sigma_globalxx(1)=sigma_xyz_bot(1,1);
sigma_globalxx(2)=sigma_xyz_mid(1,1);
sigma_globalxx(3)=sigma_xyz_top(1,1);

for i=1:n-1
    sigma_globalxx(i*3+1)=sigma_xyz_bot(1,i+1);
    sigma_globalxx(i*3+2)=sigma_xyz_mid(1,i+1);
    sigma_globalxx(i*3+3)=sigma_xyz_top(1,i+1);
end
%%Sigma yy bot mid i top juntes
sigma_globalyy(1)=sigma_xyz_bot(2,1);
sigma_globalyy(2)=sigma_xyz_mid(2,1);
sigma_globalyy(3)=sigma_xyz_top(2,1);
for i=1:n-1
    sigma_globalyy(i*3+1)=sigma_xyz_bot(2,i+1);
    sigma_globalyy(i*3+2)=sigma_xyz_mid(2,i+1);
    sigma_globalyy(i*3+3)=sigma_xyz_top(2,i+1);
end
%%Sigma xy bot mid i top juntes
sigma_globalxy(1)=sigma_xyz_bot(3,1);
sigma_globalxy(2)=sigma_xyz_mid(3,1);
sigma_globalxy(3)=sigma_xyz_top(3,1);
for i=1:n-1
   sigma_globalxy(i*3+1)=sigma_xyz_bot(3,i+1);
   sigma_globalxy(i*3+2)=sigma_xyz_mid(3,i+1);
   sigma_globalxy(i*3+3)=sigma_xyz_top(3,i+1);
end

%% Sigma xx bot mid i top juntes LOCALS
sigma_localxx(1)=sigma_123_bot(1,1);
sigma_localxx(2)=sigma_123_mid(1,1);
sigma_localxx(3)=sigma_123_top(1,1);

for i=1:n-1
    sigma_localxx(i*3+1)=sigma_123_bot(1,i+1);
    sigma_localxx(i*3+2)=sigma_123_mid(1,i+1);
    sigma_localxx(i*3+3)=sigma_123_top(1,i+1);
end
%%Sigma yy bot mid i top juntes
sigma_localyy(1)=sigma_123_bot(2,1);
sigma_localyy(2)=sigma_123_mid(2,1);
sigma_localyy(3)=sigma_123_top(2,1);
for i=1:n-1
    sigma_localyy(i*3+1)=sigma_123_bot(2,i+1);
    sigma_localyy(i*3+2)=sigma_123_mid(2,i+1);
    sigma_localyy(i*3+3)=sigma_123_top(2,i+1);
end
%%Sigma xy bot mid i top juntes
sigma_localxy(1)=sigma_123_bot(3,1);
sigma_localxy(2)=sigma_123_mid(3,1);
sigma_localxy(3)=sigma_123_top(3,1);
```

```
for i=1:n-1
   sigma_localxy(i*3+1)=sigma_123_bot(3,i+1);
   sigma_localxy(i*3+2)=sigma_123_mid(3,i+1);
   sigma_localxy(i*3+3)=sigma_123_top(3,i+1);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%%% Calcul de propietats polars, entrar salts de grau en theta
%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%

theta=1;
[Exx, Gxy, Vxy, Exxf, Gxyf, Vxyf, rot] = Apolar(sq,n,S,z,theta,h);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%%% Representacio grafica
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%

figure(1)
subplot(3,1,1),plot(sigma_globalxx,zbmt)
title('Sigma xx global');
subplot(3,1,2),plot(sigma_globalyy,zbmt)
title('Sigma yy global');
subplot(3,1,3),plot(sigma_globalxy,zbmt)
title('sigma xy global');

figure(2)
subplot(3,1,1),plot(sigma_localxx,zbmt)
title('Sigma xx local');
subplot(3,1,2),plot(sigma_localyy,zbmt)
title('Sigma yy local');
subplot(3,1,3),plot(sigma_localxy,zbmt)
title('sigma xy local');

figure(3)
subplot(1,3,1),plot(epsilon_bmt(1,:),zbmt,'+-k');
title ('deformacio epsilonxx');
subplot(1,3,2),plot(epsilon_bmt(2,:),zbmt,'+-b');
title ('deformacio epsilonyy');
subplot(1,3,3),plot(epsilon_bmt(3,:),zbmt,'+-r');
title ('deformacio epsilonxy')

figure(4)
subplot(1,3,1),polar(rot,Exx,'-b')
title ('Exx');
subplot(1,3,2),polar(rot,Gxy,'-g')
title ('Gxy');
subplot(1,3,3),polar(rot,Vxy,'-b')
title ('vxy');

figure(5)
subplot(1,3,1),polar(rot,Exxf,'-b')
title ('Exx flexio');
subplot(1,3,2),polar(rot,Gxyf,'-g')
title ('Gxy flexio');
```

```
subplot(1,3,3),polar(rot,Vxyf,'-b')
title ('vxy flexio');

bp = 200;           % Stringer width [mm]
[Carrega_critica_ref,K_x] = func_k(D11,D22,D33,D12,D13,D23,bp)
```