Trabajo Fin de Grado
Ingeniería Aeroespacial

Implementation of Special Singular Finite Elements for Cracks along Adhesive Interfaces in Mode III

Autor: Antonio Manuel Alonso Arias

Tutores: Vladislav Mantič Leščišin, Luis Arístides Távara Mendoza

Dpto. de Mecánica de Medios Continuos y Teoría de Estructuras
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2022

Trabajo Fin de Grado
Ingeniería Aeroespacial

Implementation of Special Singular Finite Elements for Cracks along Adhesive Interfaces in Mode III

Autor:

Antonio Manuel Alonso Arias

Tutores:

Vladislav Mantič Leščišin, Luis Arístides Távara Mendoza

Profesores Titulares

Dpto. de Mecánica de Medios Continuos y Teoría de Estructuras
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2022

Proyecto Fin de Carrera: Implementation of Special Singular Finite Elements for Cracks along Adhesive Interfaces in Mode III

Autor: Antonio Manuel Alonso Arias
Tutores: Vladislav Mantič Leščišin, Luis Arístides Távara Mendoza

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Acknowledgements

Scientific knowledge and engineering only make sense when they are bound to make people more free and happier. When technology improves human lives, it allows them to look beyond their own survival and to aim for discovering the world around them. Hence, they become free to make decisions by themselves, and ultimately prevent others from making decisions on behalf of them (and usually, against them). If technology has such a democratic purpose, why is it that elitism makes its way to many a technician's heart?

I want to thank those who encouraged me to maintain my thirst for knowledge amidst this impassible world. They understood my failures and corrected my errors because they saw potential in me and my generation. Among them are my tutors Mar, Luis, and Vladislav; and professors Miguel Pérez-Saborid, Cristina Arévalo, and Eva Pérez. A special thanks to Alberto, whose simulations were crucial to the research and to Alejandro Estefani for organising an Abaqus seminar. As Ludwig van Beethoven once said: 'I know no other sign of superiority than Goodness'.

Finally, I want to thank those who were also in darkness but made this world worth fighting for. The light shines in the darkness, and the darkness has not understood it.

Antonio Manuel Alonso Arias
Sevilla, 2022

Resumen

El MEF puede aplicarse a la Mecánica de la Fractura para simular tensiones en el entorno del borde de grieta; así se puede calcular el Factor de Intensificación de Tensiones. Los elementos lineales aportan tensiones constantes en su interior, por lo cual no son apropiados para captar la tendencia al infinito, requiriéndose mallas densas. Para reducir el coste computacional, se diseñan Elementos Singulares, cuyas funciones de forma y/o disposición de los nodos permite captar mejor las tensiones en torno al borde de grieta con menor número de elementos.

Aun así, un comportamiento correcto de cualquier elemento singular es muy dependiente del mallado de la pieza: forma, disposición y tamaño de los elementos alrededor de la región mallada con elementos singulares, y el tamaño relativo entre elementos y grieta entre otros. El objetivo de este trabajo es desarrollar un breve manual de uso de un nuevo Elemento Singular con singularidad logarítmica adaptado para interfases adhesivas en modo III. Como objetivo secundario, el proceso de pruebas se generaliza para extrapolarse fácilmente. Finalmente, los códigos y macros generados se adjuntan para que el proceso se aplique fácilmente a nuevos Elementos Singulares.

Abstract

FEM can be applied to Fracture Mechanics to simulate stresses in the neighborhood of a crack tip; thus, the Stress Intensity Factor can be estimated. Linear Elements bear constant tensions inside, so they are not suitable for capturing the tendency to infinity, hence requiring dense meshes. To reduce computation time, Singular Elements are designed, whose basis functions and/or nodal placing allow to better fit stresses around the crack tip with a lesser number of elements.

However, the correct behavior of any Singular Element is heavily dependent on the part meshing: shape, placing and size of the elements surrounding the singular-element-meshed region near the crack and the relative size between elements and crack among others. The purpose of this work is to develop a series of guidelines on the use of a newly programmed Singular Element with a logarithmic singularity adapted to adhesive interfaces in mode III. As a secondary purpose, the testing process is generalized so that it can be easily extrapolated. Finally, generated codes and macros are appended so that the process can be easily applied to another element.

Contents

Notation

	We will be using Einstein's notation
BVP	Boundary Value Problem
SE	Singular Element
ul	Unit of length
up	Unit of pressure
$K_{I,II,III}$	Stress Intensity Factor for modes I, II or III
$K_{IC,IIC,IIIC}$	Fracture toughness of the material for modes I, II or III
K	Generalized Intensity Factor
σ_{ij}	Cauchy stress tensor
X_{ij}	Volumetric forces affecting a solid
D	Solid domain
∂D	Solid's contour region
ε_{ij}	Strain tensor
$\mathbb{E}$	Young Modulus
σ_E	Yield strength
ν	Poisson's ratio
K	Bulk modulus
G	Shear modulus
λ	Lamé's first parameter
μ	Lamé's second parameter
∂D_t	Solid's contour region where tension is known
∂D_u	Solid's contour region where displacements are known
∂D_k	Solid's contour region where there is a relation between displacements and stresses
T	Traction vector
L_Q	Quarter-point element size
L_N	Normal element size
L_{Ni}	size of the ith normal element layer which encompasses the other (i-1)th singular elements
L_T	Transition element size
L_S	Size of the singular elements (either normal or transition elements)
L_SRZ	Size of the SRZ
u, Ψ	Displacement fields
ξ, η	Natural coordinates of an finite element
α_i	Parameters of a crown

1 The Case Study

If one must fight or create, it is necessary that this be preceded by the broadest possible knowledge.

KAREL ČAPEK

The purpose of this work is finding out the positive traits of a mesh which utilizes the Singular Elements developed by Manuel Romero in [17] for 2D cracks along adhesive boundary layers in mode III. We will focus on the traits of the elements surrounding the crack: both the singular elements, and the non-singular elements around them (which generally form one or several crowns, e.g. [34], [6]). Finally, we will figure out guidelines of size, shape and placing (among others) of the elements, which will allow us to compute a more realistic solution with the lesser number of elements possible. Following those guidelines will reduce the operational cost of solving the problem thus reducing solving time.

1.1 Hypotheses

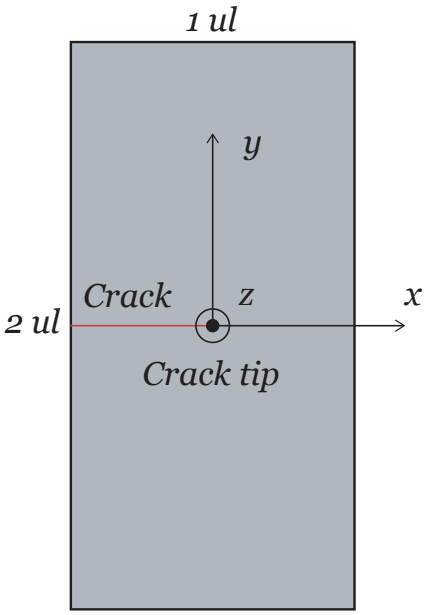


Figure 1.1 Plate and Reference System.

Our case study has the following shape and characteristics:

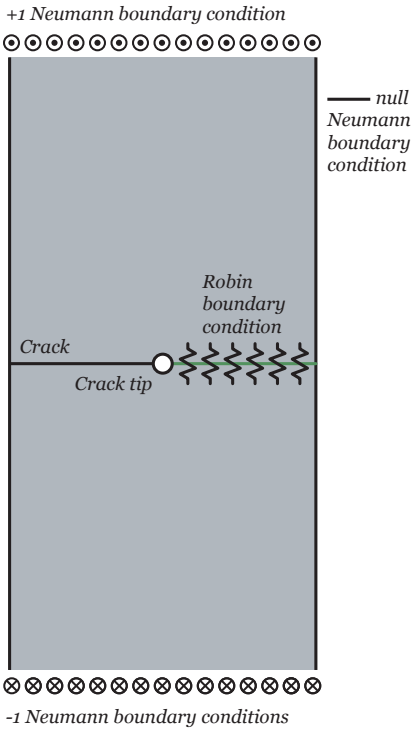
- Brittle linear elastic material with no plastification. This models a thin (orthotropic with principal directions tangential and normal to the surface) adhesive layer that joins two plates of other (isotropic and homogeneous) materials, for instance composite material laminates (of isotropic behavior). This model (and its FEM approach) has been proven valid in [22], [33] and [21].
- Body approximated by a 2D Shell of base $1ul$ and height $2ul$.
- Crack growing in the longitudinal half of the plate and expanding from one lateral edge to the center of the plate.
- The coordinate system will be as described in the diagram on the left.
- Neumann boundary conditions in the top edge (with value $1up$), bottom edge (with value $-1up$), lateral edges and open crack lips (both with null value). They model a continuous load on the top and bottom edges of the plate perpendicular to its plane.
- Robin boundary conditions in the crack growth direction, with the elastic constant (k) as a parameter. They model an adhesive interface which maintains the two parts joined at the right half but has been broken at the left half. This is equivalent to a homogeneous

and constant distribution of springs. Ultimately this relates the perpendicular stress at each pair of correlative points in the upper and lower edges with their relative displacement (see Hooke's low later).

From all of the above, we can infer:

- From the above we can infer a load skew-symmetry, hence we can study only the top half with minor changes. For instance, the half-model springs will have twice the stiffness than the whole-model springs. Stress on a node in Robin edges is equal to the elastic constant of the spring times its elongation (which we assume equal to the node displacement). The half-model elongation is half of the whole-model elongation since the origin of coordinates is in the undeformed plate, halfway between the separated sides of the plate. Hence, the half-model spring stiffness should be doubled to compensate this fact (if we want to calculate the real displacement field).
- Finally, we can also infer that the crack grows under mode III (anti-plane problem), separating the upper and lower crack lips and sides of the Robin edges. These edges are still joined together by the adhesive layer which remains intact up until the crack growth.
- Let us accept that we can define a Generalized Stress Intensity Factor K (which would not exactly be considered K_{III} due to the spring distribution) in order to compare it to the Stress Intensity Factor.

As remarked by [27], these models have a practical use on modern aviation (e.g. F18) as most composite material structures have plies that need adhesives to be joined, and even some non-composites are also bonded. Aircrafts like B787 have carbon laminate wings attached to fiber glass leading edges, and A350 has wings mainly composed of carbon composites bonded to monolithic materials that compose the flaps.



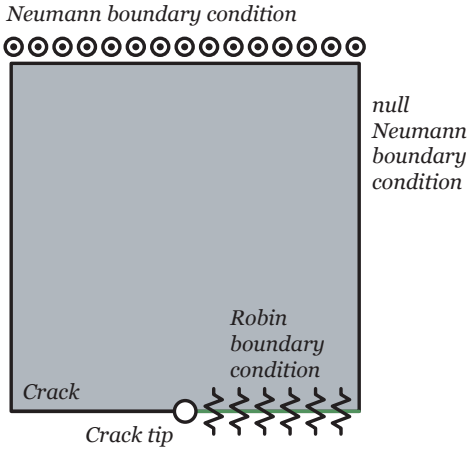


Figure 1.3 Half model boundary conditions.

Figure 1.2 Whole model boundary conditions.

As in other mode III cases, the stress tendencies in the vicinity of the crack are only dependant on the geometry of the specimen and the loads. However, unlike habitual mode III cases, in which stresses tend to infinity near the crack tip, the presence of springs make displacement (and according to Hooke's law, stresses) finite on the crack tip. More precisely, according to [4], displacements are proportional to a series of terms of the shape $r^n \log(r)$ and $r^n \log^2(r)$, which (although indeterminate for $r = 0$) has a finite limit. Its gradient,

however, will grow proportionally to $\log(r)+1$, which has a singularity for $r=0$. This type of tendency can be difficult to capture in FEM with normal means (that is, no special elements) due to the logarithmic singularity in stresses. A more thorough study of those cases can be found in [15] or [18].

1.2 Process

Small changes in the code can make the following process applicable to cases with different boundary conditions.

Also, for this study we are testing a Singular Element developed in [17] using the base of functions found in [4] [1]. Such element will allow us to properly fit our calculations to the tendency of displacements around the crack tip (employing a similar principle to the Quarter Point Element). The goal is to work out a Generalized Stress Intensity Factor for Mode III with Neumann-Robin conditions (K_{IIIN-R}) in order to determine whether the crack advances by comparing its value to the adhesive layer's fracture toughness.

However, there are still no fixed guidelines for the element usage conditions. Firstly, we ought to determine the meshing properties that would optimize the element's accuracy; that is to say, a handbook. More precisely, we will be considering how variations in the following parameters affect the FEM Outcome of the model (they will be further explained in the following chapter):

- Shape of the elements around the crack tip: depending on the direction of the elements edges we will have worse or better results since if affects the mesh quality and the number of nodes in the approaching direction.
- Number of elements around the crack tip: we will consider cases with 4, 6, 8, 12 and 24 in case a higher number of elements improves the approximation.
- Size of the elements in proportion to the crack.
- Effect of the use of a crown of non-singular elements surrounding the singular elements around the crack.
- Effect of the use of two crowns of non-singular elements surrounding the singular elements around the crack.
- Proportion between the size of singular elements and the sizes of non-singular elements composing each crown.
- Proportion between the number of singular elements and the number of non-singular elements composing each crown.
- Size of the singular-element region in proportion to the half-hexagon around it employing the optimal proportion among crown elements.
- Size of the singular-element region in proportion to the total crack length.
- Future use of singular elements (e.g., Quarter point or others) instead non-singular elements surrounding the singular elements.

The result will be considered favorable if using singular elements arranged in a certain configuration improves the result obtained without using no singular elements. That is, if a coarse or fine meshing with singular elements converges faster than a finer meshing without singular elements. Since there is no analytical solution for this precise case values, we will be taking as an exact solution the result of a convergence study with fine meshing, using N=1024 isosceles elements on each side of the plate of the same size in [22], and considered acceptable since it has been obtained by several meshing types.

Some codes, Abaqus macros, and procedures that may be useful have been developed. Even if they are just tools used along the process, they are nonetheless at the service of the University for further studies.

2 Previous knowledge

Before tackling the coding of our parametric studies, it is needed to fully comprehend our analytic and numeric tools. Firstly, we must go over Elasticity to justify the Finite Elements Method. Then, we should analyze why Singular Finite Elements help in Fracture Mechanic studies in order to develop a new element's shape functions. We will be revising our new element's shape functions deduction. Knowing this, we will justify why use conditions affect the final result of a simulation. And finally, we will analyze the code employed to implement our new element.

2.1 Elasticity Theory Equations

This sections uses the notation and formulation put forward in [8].

2.1.1 Hypotheses

- Small displacements: it allows us to identify the initial and final situation. In consequence is that tensions are produced in the deformed solid, but the force balance is applied to the undeformed geometry. It affects to the equilibrium equation.
- Small strain: it allows to consider the unitarian variations on displacement on coordinate axes as small magnitudes. In consequence, the product of displacement derivatives can be neglected in presence of the displacements. It affects the compatibility equation.
- Homogeneous material: material properties do not change inside the solid volume. That is to say, we do not have a gradient or discontinuities. It affects the material constitutive constants.
- Quasi-static load application, and material properties unaffected by time.

Afterwards we will loosen up some hypotheses to cover more study cases, specially to allow the crack tip study, which suppose a discontinuity in the medium properties (and imply plastification near the crack tip).

2.1.2 Equations

- Internal Equilibrium Equations: these equations grant that tensions inside the solid are balanced with volumetric forces affecting the solid.

$$\sigma_{ij,j} + X_i = 0 \quad \text{in} \quad D \tag{2.1}$$

- Displacement-strain relations (Saint-Venant's compatibility condition): these equations grant that the strain field is compatible with the displacement field. In other words: if the displacement field is continuous and univalued (two infinitesimaly proximal points in the undeformed situation are still infinitesimaly proximal in the deformed situation, and no two diferent points in the undeformed situation

end up in the same point in the deformed situation), then the strain field can be the solution to an elastic problem.

$$\varepsilon_{ij} = \frac{1}{2}\left(u_{i,j} + u_{j,i}\right) \quad \text{in} \quad D \tag{2.2}$$

These equations can be developed as:

$$\varepsilon_{ij,kl} - \varepsilon_{ki,jl} - \varepsilon_{lj,ki} + \varepsilon_{kl,ji} = 0 \quad \text{in} \quad D \tag{2.3}$$

And simplified as:

$$S_{pq} = e_{pik} e_{qjl} \varepsilon_{ij,kl} = 0 \tag{2.4}$$

These equations are reciprocal to (and redundant with) the Saint-Venant's condition. They grant that the strain field is compatible. That means that it can be integrated and a displacement field (continuous and univalued) can be obtained.

- Elastic Behaviour Law (or Material Constitutive Equations). This formulation is known as Generalized Hooke's Law (or flexibility equations). It relates the deformations that appear in a point of our solid to the tensions through the material's properties (only two constants are used due to the isotropic material hypothesis):

$$\varepsilon_{ij} = \frac{1+v}{E}\sigma_{ij} - \frac{v}{E}\sigma_{kk}\delta_{ij} \quad \text{in} \quad D \tag{2.5}$$

The following expressions are known as Lamé Equations (or Stiffness Equations). They connect the tension that appear in a point of the solid to the strain field. Depending on the solution strategy, we can use either Lamé or Hooke's equations.

$$\sigma_{ij} = 2G\varepsilon_{ij} + \lambda \varepsilon_{kk}\delta_{ij} \quad \text{in} \quad D \tag{2.6}$$

2.1.3 The Elastic Problem

The elastic problem is constituted by a system of 15 differential equations in coupled partial derivatives and 15 unknown variables (taking into account the stress and strain symmetry). We need a boundary condition (a given value of the displacements or stresses in a space region, usually the exterior boundaries of the body) for each variable that appears derived in the equations, and yet another more condition for each derivative of a higher order for that same variable (as it is the case for second derivatives). We can distinguish between:

- Stress Boundary Condition. Known as Neumann conditions.

$$\sigma_{ij} n_j = \bar{t}_i \quad \text{in} \quad \partial D_t \tag{2.7}$$

- Displacement Boundary Condition. Known as Dirichlet conditions.

$$u_i = \bar{u}_i \quad \text{in} \quad \partial D_u \tag{2.8}$$

- Mixed Boundary Conditions. This means that certain stress and certain displacements are known, but not both at the same time.

$$\begin{cases} T_i^{n^e} = \sigma_{ik} n_k = \bar{t}_i \\ u_j = \bar{u}_j \end{cases} \quad \text{in } \partial D_{ut} \tag{2.9}$$

- Relation Between Displacement and Stress Boundary Condition. Known as Robin Conditions. It is conceptually a generalization of a 1D spring equation.

$$T_i^{n^e} = -k_{ij} u_j \quad \text{in} \quad \partial D_k \tag{2.10}$$

As it is clear, the stiffness (the value of k) is a different value for each case. By extension, we consider this type of relation as one of the mixed conditions.

And we know the values in all the contour:

$$\partial D_t \cup \partial D_u \cup \partial D_{ut} \cup \partial D_k = \partial D \tag{2.11}$$

This will be the 6 boundary condition needed to solve the elastic problem. More precisely, they are the ones which will allow us to differentiate an elastic problem from another: our displacement, strain and stress functions must satisfy the previous equations in order to be our problem's solution, and must satisfy the boundary conditions in order to be a concrete problem's solution (by the Uniqueness theorem).

Thanks to this formulation, we are able to solve simple problems (distributed loads, simple geometries, etc.). However, it is possible to generate a method for solving the equations on a domain with more complex geometries; at the expense of losing precision at some points and solving a (probably) vast linear equation system.

2.2 Virtual Displacement Theorem

According to the lectures [25], the Virtual Displacement Theorem allows us to make an energetic analysis which is in the base of the Finite Elements Method. We start with the same hypotheses but we loosen up the isotropic medium: Compatibility Equations will not be used.

The Virtual Displacement theorem is enunciated as follows: *the only necessary and sufficient condition for the stress tensor σ_{ij} to be in equilibrium with volume forces X_i and contour forces $\bar{t}_i$ is that the work done by the stress tensor on the strain tensor be equal to the work done by external loads on displacement for every (that is, for each and every) strain field ε_{ij} compatible with its correspondent u_i associated*. It is analogue to the Principle of Virtual Work but assuming that the virtual state is not the strain field but the stress field.

Thus, hypotheses can be expressed as:

$$\varepsilon_{ij}^{\Psi} = \frac{1}{2}\left(\Psi_{i,j} + \Psi_{j,i}\right) \quad \text{in} \quad D \tag{2.12}$$

$$\Psi_i = \bar{\Psi}_i \quad \text{in } \partial D_{\Psi} \tag{2.13}$$

$$\int_D \sigma_{ij}\varepsilon_{ij}^{\Psi} dV = \int_D X_i \Psi_i dV + \int_{\partial D} t_i^c \Psi_i^c dS \quad \textit{for each and every } \Psi \textit{ with its } \varepsilon_{ij}^{\Psi} \textit{ compatible associated} \tag{2.14}$$

Then, the theorem can be mathematically expressed as:

$$\sigma_{ij,j} + X_i = 0 \quad \text{in} \quad D \tag{2.15}$$

$$\sigma_{ij} n_j^e = T_e^{n^e} = \bar{t}_i \quad \text{in} \quad \partial D_t \tag{2.16}$$

To prove this (which is needed to explain the FEM method), let us begin by expression (2.14). Additionally, let us multiply the expression (2.12) by the stress tensor, then add and subtract $\frac{1}{2}\sigma_{ij}\Psi_{i,j}$ to obtain:

$$\int_D \sigma_{ij}\varepsilon_{ij}^{\Psi} dV = \int_D \sigma_{ij} \frac{1}{2}\left(\Psi_{i,j} + \Psi_{j,i}\right) dV \tag{2.17}$$

$$\int_D \sigma_{ij}\varepsilon_{ij}^{\Psi} dV = \int_D \sigma_{ij} \left[\frac{1}{2}\left(\Psi_{i,j} + \Psi_{i,i}\right) + \frac{1}{2}\sigma_{ij}\Psi_{i,j,j} - \frac{1}{2}\sigma_{ij}\Psi_{i,j}\right] dV \tag{2.18}$$

$$\int_D \sigma_{ij}\Psi_{i,j} dV - \int_D \sigma_{ij}\left[\Psi_{i,j} - \Psi_{j,i}\right] dV = \int_D \sigma_{ij}\Psi_{i,j} dV \tag{2.19}$$

Where it has been taken into account that $[\Psi_{i,j} - \Psi_{j,i}]$ is a skew-symmetric tensor, which after being multiplied by a symmetric tensor gives a null result. We then apply again the formula of the gradient of a vector product: $(\sigma_{ij}\Psi_i)_{,j} = \sigma_{ij,j}\Psi_i + \sigma_{ij}\Psi_{i,j}$, and then we apply the Gauss divergence theorem:

$$\int_D \sigma_{ij}\Psi_{i,j} dV = \int_D (\sigma_{ij}\Psi_i)_{,j} dV - \int_D \sigma_{ij,j}\Psi_i dV = \int_{\partial D} \sigma_{ij}\Psi_i n_j^e dS - \int_D \sigma_{ij,j}\Psi_i dV \tag{2.20}$$

Which we finally introduce in the initial equation to obtain:

$$\int_{\partial D} \sigma_{ij}\Psi_i n_j^e dS - \int_D \sigma_{ij,j}\Psi_i dV = \int_D X_i\Psi_i dV + \int_{\partial D} t_i^c \Psi_i^c dS \tag{2.21}$$

Since the expression must apply (by hypotheses) to each stress field and has to be true in domain and contour, we can make equal the terms for each region. On the one hand, in the domain:

$$\text{in } D: \quad -\sigma_{ij}\Psi_i = X_i\Psi_i \tag{2.22}$$

And combining and decoupling such expression, which is valid for every Ψ, as if it was a system of equations, we can conclude that:

$$-\sigma_{ij} = X_i \tag{2.23}$$

On the other hand, in the contour:

$$\text{in } \partial D: \quad \sigma_{ij}\Psi_i n_j^e = t_i^c \Psi_i^c \tag{2.24}$$

Which we should differentiate for the tension contour and the displacement contour:

$$\text{in } \partial D_u: \quad \sigma_{ij}\Psi_i n_j^e = T_i^{n^e}\bar{\Psi}_i \tag{2.25}$$

$$\text{in } \partial D_t: \quad \sigma_{ij}\Psi_i n_j^e = \bar{t}_i\Psi_i \rightarrow \sigma_{ij} n_j^e = \bar{t}_i \tag{2.26}$$

Where it has been taken into consideration what has been stated about combination and decoupling.

2.3 Finite Elements Method

It is a approximate method to calculate the displacements inside a solid deformable body governed by expressions (2.1), (2.2), and

$$\sigma_{ij} = 2G\varepsilon_{ij} + \lambda\varepsilon_{kk}\delta_{ij} \quad \text{in } D \tag{2.27}$$

and the following boundary conditions:

$$T_i^{n^e} = \bar{t}_i \quad \text{in } \partial D_t \tag{2.28}$$

$$u_i = \bar{u}_i \quad \text{in } \partial D_u \tag{2.29}$$

$$T_i^{n^e} = -k_{ij}u_j \quad \text{in } \partial D_k \tag{2.30}$$

Beginning by the Virtual Displacements Theorem, expressions (2.15-2.16). This indicates that the only necessary and sufficient conditios for a dislacement field to be the solution of an elastic problem is for it to satisfy the integral expression for each compatible displacement field. Let us include in the expression both Constitutive Equations and Strain-Displacement relation. In matrix notation, the expression is:

$$\int_D \bar{\bar{\varepsilon}}^{\Psi^T}\bar{\bar{\sigma}} dV = \int_D \bar{\bar{\Psi}}^T \bar{\bar{X}} dV + \int_{\partial D} \bar{\bar{\Psi}}^{c^T}\bar{\bar{t}}^c dS \tag{2.31}$$

Where superindex T means trasposed. We can, for simplicity, take strain and stress tensors as pseudovectors:

$$\bar{\bar{\varepsilon}} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \gamma_{12} \\ \gamma_{13} \\ \gamma_{23} \end{pmatrix}; \quad \bar{\bar{\sigma}} = \begin{pmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_{12} \\ \sigma_{13} \\ \sigma_{23} \end{pmatrix}; \quad \text{with } \gamma_{ij} = 2\varepsilon_{12} \tag{2.32}$$

Now let is introduce the Constitutive Law in the Theorem's equation. Matrices dimensions are indicated as subindices:$\sigma_{6\times1} = D_{6\times6}\varepsilon_{6\times1}$. D being:

$$D = \begin{bmatrix} 2G+\lambda & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & 2G+\lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & 2G+\lambda & 0 & 0 & 0 \\ 0 & 0 & 0 & G & 2G+\lambda & 2G+\lambda \\ 0 & 0 & 0 & 2G+\lambda & G & 2G+\lambda \\ 0 & 0 & 0 & 2G+\lambda & 2G+\lambda & G \end{bmatrix} \tag{2.33}$$

Hence the VDT can be expressed as:

$$\int_D \bar{\bar{\varepsilon}}^{\Psi^T} \bar{\bar{D}} \bar{\bar{\varepsilon}}^{\Psi} dV = \int_D \bar{\bar{\Psi}}^T \bar{\bar{X}} dV + \int_{\partial D} \bar{\bar{\Psi}}^{c^T} \bar{\bar{t}}^c dS \tag{2.34}$$

And using a strain-displacement relation such as $\varepsilon_{6\times1} = B_{6\times3}u_{3\times1}$, with B:

$$B = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 \\ 0 & \frac{\partial}{\partial y} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} & 0 \\ \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} \\ 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial y} \end{bmatrix} \tag{2.35}$$

Thus, using Ψ as displacement field, the VDT can be expressed as:

$$\int_D \bar{\bar{\Psi}}^T \bar{\bar{B}}^T \bar{\bar{D}} \bar{\bar{B}} \bar{\bar{\Psi}} dV = \int_D \bar{\bar{\Psi}}^T \bar{\bar{X}} dV + \int_{\partial D} \bar{\bar{\Psi}}^{c^T} \bar{\bar{t}}^c dV \tag{2.36}$$

This way, when introducing the Behaviour Law on the displacement field, and introducing Compatibility Equations, we can obtain our problem's solution from this expression (subject to boundary conditions).

Now let us decompose our movement field in each direction as the addition of N approximating functions times its respective intensity (we can imagine this as separating each possible movement in N possibilities whose linear combination can result in every displacement field possible):

$$|x| = \begin{cases} u_1 = \sum_{i=1}^N a_i \phi_i \\ u_2 = \sum_{i=1}^N b_i \phi_i \\ u_3 = \sum_{i=1}^N c_i \phi_i \end{cases} \tag{2.37}$$

Said functions can be chosen as full support or small support. Full support can be, for instance, polynomials ($\phi_i = x^i$). Small support can have the following shape:

$$|x| = \begin{cases} 1 \text{ in } P_i \\ 0 \text{ in } P_j \; \forall \, i \neq j \end{cases} \tag{2.38}$$

To go on, we need to make some heuristic decisions (which also define what the FEM is). These are: taking the same approximating functions for each three displacement components, and taking the same number of functions for the u_1, u_2 and u_3 series. Due to this last approximation, we have restricted the displacement that the solid can do: it only will be able to do movements defined by linear combination of approximant functions ϕ_i. That means that we have gone from an infinite number of degrees of freedom (as a continuous solid) to just $3n$ degrees of freedom (as an approximate solid). Then, we have an equation with $3n$ unknown variables, but this equation holds for every displacement field. Hence, we can 'project´ all these fields in our possible fields of the discrete solid (defined by the combinations of linear functions).

In matrix notation, the change of variable results in: $\bar{\bar{u}}_{3\times 1} = \bar{\bar{\phi}}_{3\times 3n}\bar{a}_{3n\times 1}$. Which, developed, can be expressed as:

$$\bar{\bar{u}} = \begin{bmatrix} \phi_1 & 0 & 0 & \phi_2 & 0 & 0 & & \phi_3 & 0 & 0 \\ 0 & \phi_1 & 0 & 0 & \phi_2 & 0 & \dots & 0 & \phi_3 & 0 \\ 0 & 0 & \phi_1 & 0 & 0 & \phi_2 & & 0 & 0 & \phi_3 \end{bmatrix} \begin{pmatrix} a_1 \\ b_1 \\ c_1 \\ a_2 \\ b_2 \\ c_3 \\ \vdots \\ a_n \\ b_n \\ c_n \end{pmatrix} \tag{2.39}$$

Finally, we have to solve the system:

$$\left| \int_D \bar{\bar{\Psi}}^T \bar{\bar{B}}^T \bar{\bar{D}} \bar{\bar{B}} \bar{\bar{\phi}} dV \right| \bar{\bar{a}} = \int_D \bar{\bar{\Psi}}^T \bar{\bar{X}} dV + \int_{\partial D} \bar{\bar{\Psi}}^{c^T} \bar{\bar{t}}^c dS \tag{2.40}$$

The system has $3n$ unknown variables (the components of vector $\bar{a}$) so we will need $3n$ components of $\bar{\bar{\Psi}}$ to have $3n$ equations.

$$\bar{\bar{\Psi}}_{3\times 3n} = \begin{bmatrix} \Psi_1 & 0 & 0 & & \Psi_n & 0 & 0 \\ 0 & \Psi_1 & 0 & \dots & 0 & \Psi_n & 0 \\ 0 & 0 & \Psi_1 & & 0 & 0 & \Psi_n \end{bmatrix} \tag{2.41}$$

The matrix system is $k_{3n\times 3n}a_{3n\times 1} = F^D_{3n\times 1} + F^{\partial D}_{3n\times 1}$. Which can be solved multiplying on both sides by k^{-1}. This process imposes the VDT in the discrete solid (that is, the one with possible displacements limited to $3n$), but the result need not be be true for the real solid (which has infinite possible displacements).

Up till now, this has been the general approximation method. The Finite Elements Method needs, in addition, two more heuristics.

Firstly, to select the projection functions Ψ and the approximation functions ϕ will be considered identical: $\Psi_i = \phi_i$, so that the stiffness matrix k expression will be $\int_D \bar{\bar{\Psi}}^T \bar{\bar{B}}^T \bar{\bar{D}} \bar{\bar{B}} \bar{\bar{\Psi}} dV$ which becomes a symmetric matrix. This have positive consequences: less time to generate the matrix, less storing space needed, and less time to solve the equation system.

Secondly, for ϕ_i (and by extension for Ψ_i) will be taken as small support functions: applicable to every possible domain, and easily adapted to every solution (allowing further improvements). This leads to the total displacements in a point to be directly obtained from the equation system. That means, the displacement from the node m only influences the function Ψ_m, therefore, solution a_m will be the node m's displacement. This explains why k is called stiffness matrix in the domain, and the system can finally be written expressed as:

$$ku = F \tag{2.42}$$

The system can be understood as: a matrix k_{ij} which relates each a_j to F_i. It is easy to understand F_i as the forces that should be applied to each node i to provoke a unitary displacement in node j maintaining a null displacement in the remaining nodes. The type of function which we choose makes many terms in k to be null, and those who are not, are defined for a small fraction of the domain. This is due to the fact that functions only have a common support (that means, both have a not-zero value) when said functions are defined in adjacent nodes (that means, with a common edge). If they do not have a common edge, then on or both of them will have zero value.

Hence the computing of the correspondent term of the stiffness matrix will have zero value. In consequence, the stiffness matrix will be a band matrix. This does not mean that only the diagonal will have nonzero values: there will be upper and lower bands, but most of their terms will be null. As noted in [14] The bandwidth will be approximately half the maximal difference between two global nodes inside the same element. Minimizing this would reduce the bandwidth and potentially, the solution time of the system. In order to improve it, Sanz-Herrera will use Cuthill-McKee algorithm, as we will see later.

To impose boundary conditions, it is enough to note that the independent term is the load vector, the solution vector are the displacements (some of them known are already known by boundary conditions) and Robin conditions suppose additional equations that complete the system (am additional relation between

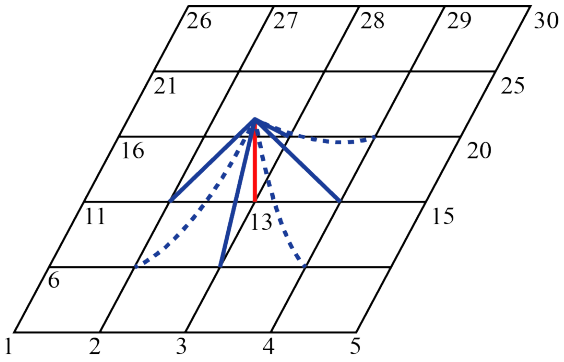


Figure 2.1 Example of small support functions in node 13.

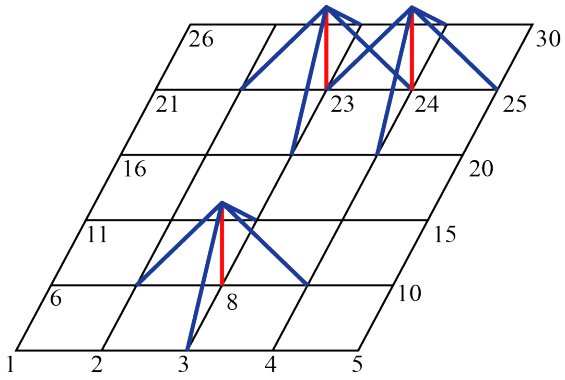


Figure 2.2 Nodes 8 and 23 do not share common space, hence the correspondent integral will have null value, while nodes 23 and 24 share an element region.

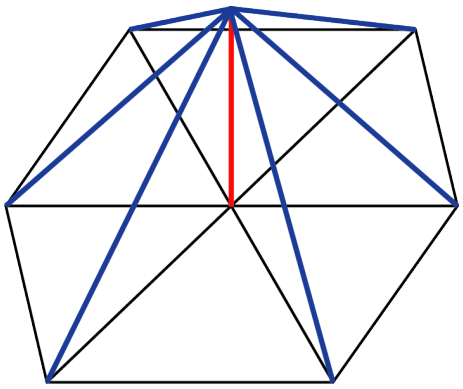


Figure 2.3 CST element, which is triangular, can be used to cover with enough goodness of fit every domain.

1	2	3	4	5	6
7 13	8 14	9 15	10 16	11 17	12 18

1	2	3 9	4 10	5 11	6
7 16	8 17	12 18	13 19	14 20	15 21

Figure 2.4 Top: uncracked solid. Bottom: cracked solid, with twice the number of nodes on the crack.

displacements and forces in a node) and substitute the unnecessary equations of the original system (that means, a new equation in the stiffness matrix). In all off these cases we would be altering the system but having a new determined compatible one.

2.4 Finite Element Method on Fracture Mechanics

Information presented as in [26] The FEM can be applied to fracture mechanics. More precisely, we can model a solid with a crack inside of it. The crack is defined as a surface along which the displacement field is discontinuous. It will follow two rules: elements boundaries must coincide with the crack (that means, no element can have a crack inside its domain), and the solid must be discretised in a way that there are different nodes on each side of the crack. These are not linked between themselves, and are not the same: they only coincide in the undeformed situation. That is to say, we have twice the number of nodes on the crack than we would have without the crack. This does not prevent us from applying symmetry if the solid is infinite or the crack is on its middle section. This can be seen in the figure 2.3.

We will be assuming LEFM hypotheses, acceptable for brittle materials. If the method is applied straightforwardly, nodal displacement will be linear inside an element, so that deformations (and by extension tensions) will be constant inside the element. In order to correctly capture the tendency of σ_y tensions in the neighborhood of the crack tip (keep in mind that theoretically it tends to infinity), the mesh must be considerably fine. Therefore, max σ_y will depend on the mesh, and will only be considered valid if, after several iterations diminishing the element's size, its value converges. These values can be compared to yield strength σ_e to determine if there is plastification, or be used to calculate the stress intensity factor K_n to

compare it to fracture toughness of the material in such mode K_c (in order to determine whether the crack grows). It can also be used to calculate and compare G_N to G_C.

Furthermore, the plastification avoids tensions to grow to infinity in the plastified region as in Irwin's crack ([23]). There will be a zone of validity in which tensions perpendicular to the crack apparently tend to infinity, but that value will never be reached to. To simulate this effect, Robin conditions can be modeled in the advance of the crack. This way, since displacements are finite, the appearing stresses will also be finite. Hence we can compute in a more realistic way K_N (a global trend, detailed by [10]). This method, implemented in FEM, can be applied to any crack to achieve best convergence and more precise results. This applies to any crack, however the stiffness constant will be comparatively higher in cases without springs, and the advance direction should be previously known. In those cases, however, the approach of LEBIM+FFM (with coupled stress and energy criteria) will be more suitable, as shown in [24].

2.5 Singular Elements in Fracture Mechanics

2.5.1 Analytic solution

Following a logic similar to that of [9], we will be using the Finite Element Method for Linear Elasticity (fragile materials), and as stated: small deformations, homogeneous and isotropic materials, quasi-static processes and 2D problem (knowing that there are 3D generalisations). We sill study the criticality of a fixed crack without studying propagation mechanisms. We will consider FEM in standard formulation (not XFEM).

The efficiency of this method to capture Westergaard's solution ([12], [2], [7]) have proven to be effective and do not have high computational cost.

The goal is to compute K_i, which depends on the geometry and length of the crack and on the loads. In a simple case (only Mode I, II or III), we will compare said value with fracture toughness (K_{IC}, K_{IIC} or K_{IIIC}) which is a material property. From these values, for a mixed case we must define a criterion based on elasticity with an expression like:

$$f\left(K_I, K_{II}, K_{III}\right) \leqslant f_C \tag{2.43}$$

However, the method has a limitation. For a crackless solid:

$$\left\|\bar{u}-\bar{u}^h\right\|_E = O\left(h^P\right) \qquad \text{If } \|\bar{u}\|_{H^{P+1}} < +\infty \tag{2.44}$$

For a solid with a crack, the displacement field has a $H^1 - regularity$, so the compute will be less precise:

$$\left\|\bar{u}-\bar{u}^h\right\|_E = O\left(h^{\frac{1}{2}-\eta}\right) \quad \text{with } \eta > 0 \tag{2.45}$$

Assuming a high regularity in the displacement field, the solution and all its derivatives until order $P+1$ are square-integrable functions. By using an interpolation of a higher order, we can achieve a better solution in the continuous solid, but it is not the case if the solution is irregular, as a cracked solid.

To better comprehend the result's precision we can use the error's norm. The stress field has a singularity near the crack tip, and the stress limit will be obtained thanks to extrapolation with several FEM values.The singularity problem is that FEM can not reproduce the tendency to infinity: the finer the mesh, the higher the stresses. With standard elements, at least, the singularity can not be captured.

To mathematically illustrate this, let us analyze Mode III tensions near the crack tip (where $r \to 0$).

$$\sigma_{\theta z} = \frac{K_{III}}{\sqrt{2\pi r}} \cos\frac{\theta}{2} \tag{2.46}$$

$$u_z = \frac{2K_{III}}{G}\sqrt{\frac{r}{2\pi}} \sin\frac{\theta}{2} \tag{2.47}$$

Comparing in a graphic FEM and analytic results, we can distinguish three regions:

In the first region, FEM results are dependent on the mesh, so they arrive to a certain finite value. In the third region, FEM gives a realistic solution, but asymptotic solution loses validity. Only in the second region the solution is dominated by the singularity and FEM gives a realistic solution. This will be the extrapolatable region. To better capture this behaviour, we will use Singular Elements.

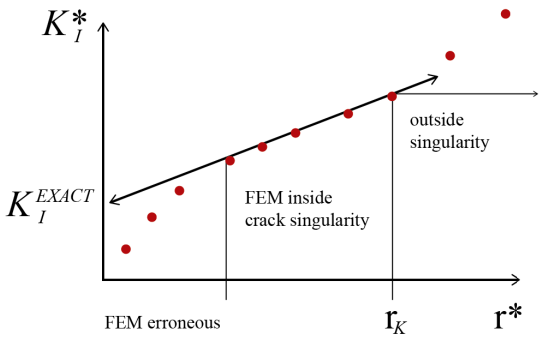


Figure 2.5 Only the middle region gives a good approximation for the stress intensity factor.

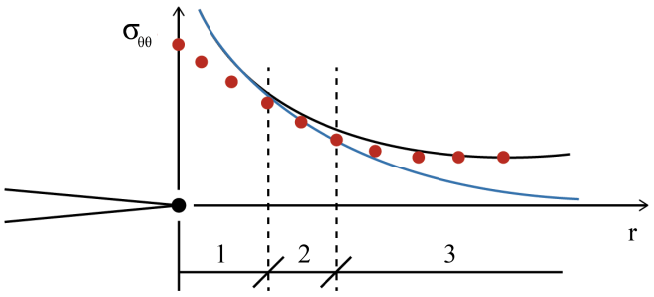


Figure 2.6 Only in the second region FEM gives a realistic solution and the tensions are dominated by the singularity.

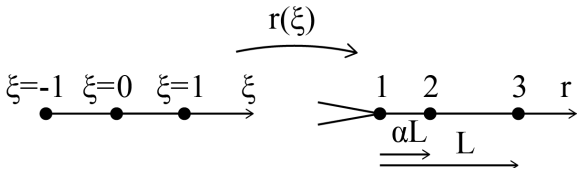


Figure 2.7 Isoparametric element and change of variable to real coordinates.

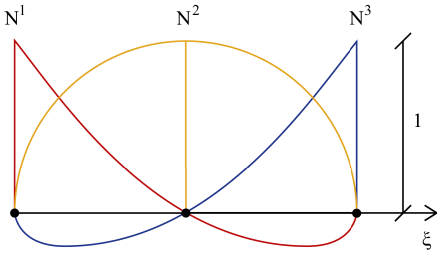


Figure 2.8 Shape functions.

2.5.2 Numeric solution

Let us consider an isoparametric quadratic element in one direction, with three interpolation functions:

$$\begin{cases} N^1(\xi) = \frac{1}{2}\xi(\xi-1) \\ N^2(\xi) = \frac{1}{2}\xi(\xi+1) \quad \text{in } \partial D_{ut} \\ N^1(\xi) = 1-\xi^2 \end{cases} \tag{2.48}$$

Which allow us to interpolate displacements and coordinates:

$$u(\xi) = \sum_{a=1}^{3} N^a(\xi)\, u^a = u^3 + \frac{1}{2}\left(u^2 - u^1\right)\xi + \left[\frac{u^1+u^2}{2} - u^3\right]\xi^2 \tag{2.49}$$

$$r(\xi) = \sum_{a=1}^{3} N^a(\xi)\, r^a = \alpha L + \frac{1}{2}L\xi + \left(\frac{1}{2} - \alpha\right)L\xi^2 \tag{2.50}$$

When $\alpha = 1/2$, $r = \frac{L}{2}(1+\xi) \rightarrow \xi = \frac{2r}{L} - 1$, so after substituting in $u(\xi)$ we obtain a polynomic quadratic function.

When $\alpha = 1/4$, $r = \frac{L}{4} + \frac{L}{2}\xi + \frac{L}{4}\xi^2 = \frac{L}{4}(1+\xi)^2 \rightarrow \xi = 2\sqrt{\frac{r}{L}} - 1$, so after substituting we obtain:

$$u(r) = u^1 + \left(4u^3 - 3u^1 - u^2\right)\sqrt{\frac{r}{L}} + 2\left(u^1 + u^2 - 2u^3\right)\frac{r}{L} \tag{2.51}$$

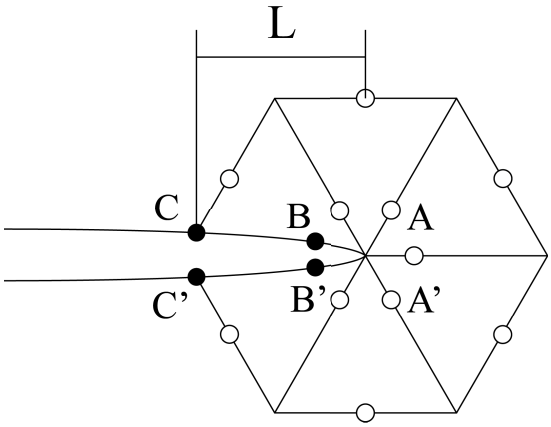


Figure 2.9 Quarter point rosette configuration.

$$\varepsilon(r) = \frac{du}{dr} = \left(2u^3 - \frac{3}{2}u^1 - \frac{1}{2}u^2\right)\frac{1}{\sqrt{Lr}} + \frac{2\left(u^1 + n^z - 2u^3\right)}{L} \tag{2.52}$$

The displacement behavior is proportional to $\sqrt{r}$ when obtained with $\alpha = 1/4$. The first term is constant, and the third term is negligible when r tends to zero. Therefore, displacement calculated with this element will be more appropriate for crack tips, in which the displacement near the crack tip behave as follows (particularised for Mode I, but analogue for the others):

$$u_\theta(r,\theta) \approx \frac{K_I}{2\mu}\sqrt{\frac{r}{2\pi}}\sin\left(\frac{\theta}{2}\right)\left[\kappa + 1 - 2\cos^2\frac{\theta}{2}\right] \tag{2.53}$$

Then, if we mesh around the crack tip with a rosette pattern, we will easily capture the behaviour in all directions. This case, however, is adapted for a Mode I or Mode II case with Neumann-Dirichlet conditions. In the following section, we will be adapting the element to our case. Here we are trying to prove that adapting the element surrounding the crack tip can be helpful to achieve a better result.

2.6 Alberty-Carstensen-Funken Finite Element Method Implementation

Before implementing an Abaqus subroutine for a user-defined element, it may be more simple to use an open-source FEM code instead of a black-box commercial code. However, the meshing algorithm is presumably more stable in commercial programs. The employed tool will be Manuel Romero's modification of the short finite element implementation by Alberty et al [3] redesigned by Sanz Herrera [5]. Originaly , the code calculates a numerical solution U which approximates the solution u to the 2D Laplace problem with mixed boundary conditions (Neumann and/or Dirichlet) and no singular elements.

Firstly, let us explain why shape function influence the stiffness matrix. Assuming a Galerking discretisation of the problem, let $(\eta_1,...,\eta_N)$ be a basis of the dimensional space S, and $(\eta_{i1},...,\eta_{iM})$ be a basis of S_D. These are used in a slightly different terminology to define the integral for the local entry of the stiffness matrix. The A_{jk} coefficient matrix (stiffness matrix) of this discretisation needs to be integrated in the domain of the element, and its value (as seen before) depends on approximating functions. Also the right-hand matrix, which models forces, needs to be projected on our element's functions:

$$\begin{aligned} k_{local} = A_{jk} &= \sum_{T\in\mathcal{T}}\int_T \nabla\eta_j\cdot\nabla\eta_k \,\mathrm{d}x, \\ Flocal = b_j &= \sum_{T\in\mathcal{T}}\int_T f\eta_j \,\mathrm{d}x + \sum_{E\subset\Gamma_N}\int_E g\eta_j \,\mathrm{d}s - \sum_{k=1}^{N} U_k \sum_{T\in\mathcal{T}}\int_T \nabla\eta_j\cdot\nabla\eta_k \,\mathrm{d}x. \end{aligned} \tag{2.54}$$

Developing the terms using $\eta_j(x,y) = \varphi_j\left(\Phi_T^{-1}(x,y)\right)$ results in the following matrix entry:

$$\begin{aligned} M_{jk} &:= \int_T \nabla\eta_j(x,y)\cdot\nabla\eta_k(x,y)\mathrm{d}(x,y) \\ &= \int_T \nabla\left(\varphi_j\circ\Phi_T^{-1}\right)(\Phi_T(\xi,\zeta))\left(\nabla\left(\varphi_k\circ\Phi_T^{-1}\right)(\Phi_T(\xi,\zeta))\right)^{\mathrm{T}}|\det D\Phi_T|\,\mathrm{d}(\xi,\zeta) \\ &= \det(D\Phi_T)\int_T \nabla\varphi_j(\xi,\zeta)\left((D\Phi_T)^{\mathrm{T}}D\Phi_T\right)^{-1}(\nabla\varphi_k(\xi,\zeta))^{\mathrm{T}}\,\mathrm{d}(\xi,\zeta) \end{aligned} \tag{2.55}$$

Likewise, the volume forces f and contour stresses can be approximated in the element's center.

As stated before, to incorporate Neumann conditions, we change the value of g. To incorporate Dirichlet conditions, we can divide our system:

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{12}^{\mathrm{T}} & A_{22} \end{pmatrix}\cdot\begin{pmatrix} U \\ U_{\mathrm{D}} \end{pmatrix} = \begin{pmatrix} b \\ b_{\mathrm{D}} \end{pmatrix} \tag{2.56}$$

A process which is used to remove the unnecessary part of stiffness matrices thanks to the boundary conditions. Since the Dirichlet node's displacements are known, the equations can be rewritten as: $A_{11}\cdot U = b - A_{12}\cdot U_{\mathrm{D}}$.

Robin condition application is developed by Manuel Romero, being implemented similarly to Neumann conditions, however f being computed as follows:

$$\int_T f dl = \ length\ \cdot \int_T f d\xi \tag{2.57}$$

2.7 Analytic Solution of a Neumann-Robin 180º corner

In [4] the solution (vertical displacements) of a corner with angle ω and Robin conditions in one of its faces is found as an asymptotic series for plane and antiplane cases (the study theorised an infinite plate specimen, that is, only the proximity of the crack). More precisely, for Mode III and $\omega = \pi$ cases. The demonstration departs from the Laplace equation, since vertical displacement is a harmonic function, and goes on imposing the boundary conditions for each case. Initially a Neumann-Neumann BVP case is solved to compute the main terms in the series $w_j^{(k)}(r,\theta)$. After that, shadow terms $w_j^{(0)}(r,\theta)$ (which are used to reduce the error given by said approximation) are calculated by solving recursive BVP's. Each shadow term is obtained thanks to the following expression, in which we can see how the upper boundary of the summation k coincides with the power of the logarithmic expression, thus having to augment it to increase the order of the series:

$$w_j^{(k)}(r,\theta) = \sum_{l=0}^{k}\left[a_{j,k}^{(l)}\cdot Im\left\{z^{\lambda_j+k}log^l z\right\} + b_{j,k}^{(l)}\cdot Re\left\{z^{\lambda_j+k}log^l z\right\}\right] \tag{2.58}$$

For our case, $\omega = \pi$ is an actual critical angle, hence an infinite series is needed to make the error completely vanish. Coefficients a must be zero as imposed by Neumann boundary condition, and b coefficients are obtained by imposing the Robin boundary condition (with the original formulation in the paper being as follows).

$$\frac{1}{r}\sum_{l=0}^{k}\frac{\partial}{\partial\theta}\left[b_{j,k}^{(l)}Re\left\{z(r,\theta)^{\lambda_j+k}log^l z(r,\theta)\right\}\right]_{\theta=\pi} = -\gamma_z\sum_{l=0}^{k-1}b_{j,k}^{(l)}Re\left\{z(r,\theta)^{\lambda_j+k}log^l z(r,\theta)\right\} \tag{2.59}$$

When expressing the equation in real variables, since it holds $\forall r$, we can compare terms with the same $log^m r$. Hence we end up with a system of equations of the shape $\bar{\bar{M}}_{j,k}\bar{B}_{j,k} = \bar{G}_{j,k}$ with $\bar{\bar{B}}$ being the vector of coefficients $[b_{j,k}^{(0)}, b_{j,k}^{(1)}, \ldots b_{j,k}^{(k)}]^T$, $\bar{\bar{G}}$ being the righ-side coefficients vector $[G_{j,k}^{(0)}, G_{j,k}^{(1)}, \ldots G_{j,k}^{(k)}]^T$ with $G_{j,k}^{(m)}(\pi) = \gamma_z\sum_{l=m}^{k-1}b_{j,k}^{(l)}\binom{l}{m}\pi^h Re\left\{i^h\right\}$ and $G_{j,k}^{(k)}(\pi) = 0$ (since from one shadow term to the following, in this case the two last ones, there is always an extra logarithmic term), and $\bar{M}$ with the following structure:

$$M_{m,l} = \binom{l}{m}(h\pi^{h-1}Re\left\{i^h\right\} + \pi^h(j+k-1))Re\left\{i^{h+1}\right\}),\ \text{if}\ \ m < l \tag{2.60}$$

$$M_{m,l} = 0, \text{ if } m \geq 1 \tag{2.61}$$

From here, coefficients can be found, and the displacement in the vicinity of the crack fully characterised:

$$b_{j,k}^{(k)} = \frac{G_{j,k}^{(k-1)}}{M_{k-1,k}} \tag{2.62}$$

$$b_{j,k}^{(l)} = \frac{1}{M_{l-1,l}}[G_{j,k}^{(l-1)} - \sum_{n=l+1}^{k} M_{l-1,n} \cdot b_{j,k}^{(n)}], \text{ if } 0 < l < k \tag{2.63}$$

Using only the first two shadow terms ($S_j = 2$) we can arrive to a trade-off between the computation time (which is incremented with each shadow term) and a sufficiently good approximation. The same research shows that using four shadow terms is more than enough to capture displacement tendencies. Firstly, we compute the Neumann-Neumann solution and then we make a change of variable $\theta = \pi - \hat{\theta}$, thus obtaining $w_j^{(0)}(r, \widehat{\theta}) = r^{j-1}\cos[(j-1)(\pi - \widehat{\theta})]$. Then the following terms are computed as explained before:

$$\begin{aligned} w_j^{(1)}(r,\theta) &= \frac{-\gamma_z \cos[j\pi] r^j}{j\pi}((\pi - \widehat{\theta})\sin[j\widehat{\theta}] + \cos[j\widehat{\theta}]\log r) \\ w_j^{(2)}(r,\widehat{\theta}) &= \frac{-\gamma_z^2 r^{1+j}\cos[(1+j)\pi]}{j(j+1)\pi^2}\left[W_1(r,\widehat{\theta}) + W_2(r,\widehat{\theta})\right] \end{aligned} \tag{2.64}$$

with

$$\begin{aligned} W_1(r,\widehat{\theta}) &= \frac{1}{(j+1)}((\pi - \widehat{\theta})\sin[(1+j)\widehat{\theta}] + \cos[(1+j)\widehat{\theta}]\log r) \\ W_2(r,\widehat{\theta}) &= \frac{1}{2}\left[\cos[(1+j)\widehat{\theta}]\left((\pi - \widehat{\theta})^2 - \log^2 r\right) - 2(\pi - \widehat{\theta})\log r \sin[(1+j)\widehat{\theta}]\right] \end{aligned} \tag{2.65}$$

2.8 Our element

Making an analogy with the quarter-point element [9], we can obtain an element able to capture the behaviour described by the previous formula. Furthermore, to model the whole problem, that is, with the loads in the upper part of the shell as Neumann conditions, we can establish an analogy between Fourier's heat problem (of a heat focus touching the plate) and ours, since both problems, antiplane and heat, have analogous equations ([30], [31], [32]). This is the justification for using a FEM software to analyze the antiplane problem, that is what he started working out. Moreover, in this case temperature is analogue to displacements u, Dirichlet conditions are analogue to a heat focus at a certain temperature, stresses are analogue to heat flow, Robin conditions to convection flows, and Neumann conditions to a prescribed heat flow. We can employ Jimenez's solutions ([27], [4]), which have a different expression for each aperture angle of the shell, to refine an element in order to fit the displacements near the crack tip in the desired Mode III. The basis for the 180 case, as seen before, has a spatial base of the following functions: l, r, $ln(r)$. Using the algorithmic method stated by [13], Manuel Romero was able to obtain the following set of shape functions throughout iterations by using Lagrange's Polynomials. It was worked out as a 1D element, whose shape functions caught the displacement tendencies in the ξ direction, with the following result:

$$\begin{aligned} N_3(r) &= \frac{r[\ln(1/2) - \ln(r)]}{\ln(1/2)} \\ N_1(r) &= 1 - 2r + \frac{r[\ln(1/2) - \ln(r)]}{\ln(1/2)} \\ N_2(r) &= 2r - 2\frac{r[\ln(1/2) - \ln(r)]}{\ln(1/2)} \end{aligned} \tag{2.66}$$

Then, the functions had to be adapted into the reference 2D space with an $\eta - linear$ behaviour, and for six-node elements, as shown in the following table.

N_{i_u}	$\frac{\partial N_i}{\partial \xi}$	$\frac{\partial N_i}{\partial \eta}$	i
$(1-\eta)\left(1-\xi-\xi\frac{\ln(\xi)}{\ln(1/2)}\right)$	$(1-\eta)\left(-1-\frac{1}{\ln(1/2)}-\frac{\ln(\xi)}{\ln(1/2)}\right)$	$-\left(1-\xi-\xi\frac{\ln(\xi)}{\ln(1/2)}\right)$	1
$(1-\eta)\left(2\xi\frac{\ln(\xi)}{\ln(1/2)}\right)$	$2(1-\eta)\left(\frac{\ln(\xi)}{\ln(1/2)}+\frac{1}{\ln(1/2)}\right)$	$-\left(2\xi\frac{\ln(\xi)}{\ln(1/2)}\right)$	2
$(1-\eta)\left(\xi-\xi\frac{\ln(\xi)}{\ln(1/2)}\right)$	$(1-\eta)\left(1-\frac{\ln(\xi)}{\ln(1/2)}-\frac{1}{\ln(1/2)}\right)$	$-\left(\xi-\xi\frac{\ln(\xi)}{\ln(1/2)}\right)$	3
$\eta\left(\xi-\xi\frac{\ln(\xi)}{\ln(1/2)}\right)$	$\eta\left(1-\frac{\ln(\xi)}{\ln(1/2)}-\frac{1}{\ln(1/2)}\right)$	$\left(\xi-\xi\frac{\ln(\xi)}{\ln(1/2)}\right)$	4
$\eta\left(2\xi\frac{\ln(\xi)}{\ln(1/2)}\right)$	$2\eta\left(\frac{\ln(\xi)}{\ln(1/2)}+\frac{1}{\ln(1/2)}\right)$	$\left(2\xi\frac{\ln(\xi)}{\ln(1/2)}\right)$	5
$\eta\left(1-\xi-\xi\frac{\ln(\xi)}{\ln(1/2)}\right)$	$\eta\left(-1-\frac{1}{\ln(1/2)}-\frac{\ln(\xi)}{\ln(1/2)}\right)$	$\eta\left(1-\xi-\xi\frac{\ln(\xi)}{\ln(1/2)}\right)$	6

(2.67)

After obtaining the functions, they can be collapsed into a triangular element in the real space by making the coordinates of two of its nodes equal, which will affect the element's behaviour. The process is an analogous to Barsoum's process [6]. It can be illustrated as in figure 2.7.

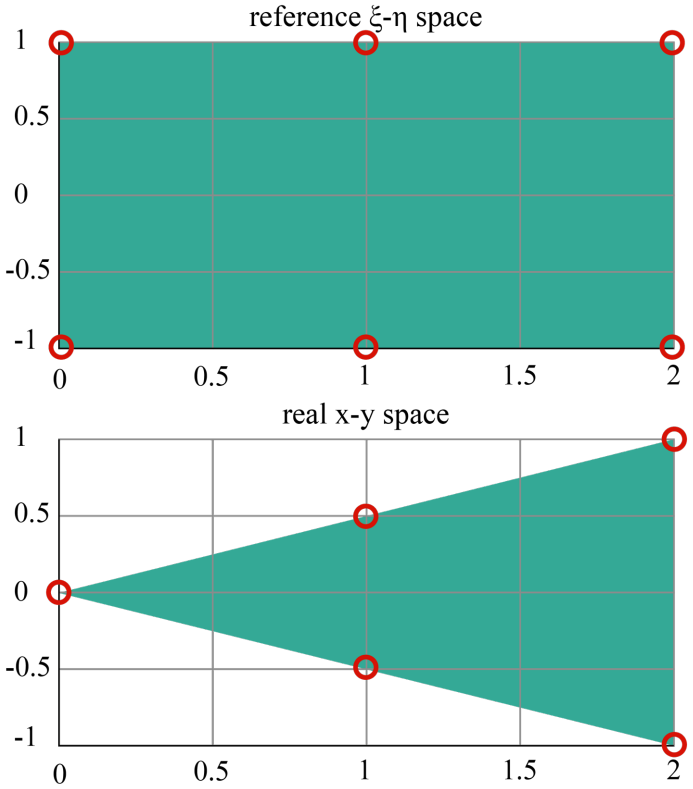


Figure 2.10 Collapse of the 2D square element into a 5 node triangle.

As a result of the collapse of the nodes, shape functions number 1 and 6 must fulfill:

$$[N_1(\xi,\eta)+N_6(\xi,\eta)]_{\xi-\eta\ space}=[N_1(x,y)]_{real\ space} \tag{2.68}$$

Finally, to integrate each entry of the stiffness matrix, the process is analogue to that of turning a 4-noded element into a triangular one. The jacobian is chosen to affect the nodes needed, not affecting the quadratic-behaviored ones; and choosing to make equal the displacements of nodes 1 and 6 (as implied before). We end up with the following equation:

$$\left(K_{ij}\right)_{5x5} = G\left(\int_0^1\int_0^1 \begin{pmatrix} \frac{\partial(N_1+N_6)}{\partial\xi} & \frac{\partial(N_1+N_6)}{\partial\eta} \\ \frac{\partial N_2}{\partial\xi} & \frac{\partial N_2}{\partial\eta} \\ \vdots & \vdots \\ \frac{\partial N_5}{\partial\xi} & \frac{\partial N_5}{\partial\eta} \end{pmatrix} \left([J]^{-1}\right)^T [J]^{-1} \begin{pmatrix} \frac{\partial(N_1+N_6)}{\partial\xi} & \cdots & \frac{\partial N_5}{\partial\xi} \\ \frac{\partial(N_1+N_6)}{\partial\eta} & \cdots & \frac{\partial N_5}{\partial\eta} \end{pmatrix} |J| d\xi d\eta\right)_{5x5} \tag{2.69}$$

Stresses inside the element and the Robin matrix can be both obtained in a relatively similar manner. The integration was done in Wolfram's Mathematica. And the whole result is added in the MATLAB code, in its own sub-section of the main program. Thanks to this, we can apply the Sanz-Herrera's rendition of Alberty-Carstensen-Funken code to our analysis changing the stiffness matrix's entries for our singular element equations.

Lastly, what utterly motivated my work was the fact that the last of Romero's studies did meet the expectations (that is, his element converged faster than CST's) even if his elements were considerable deformed due to the meshing. This may have led to a worsening of the results. However, even with such drawback, results were improved. The question is now clear: can we make convergence faster with different kinds of meshing? That is to say, it is expected that different configurations can improve current conclusions, but firstly we need to know what are the beneficial traits of a mesh. Once they are identified, the study can be remade and presumably, it will show higher convergence rate.

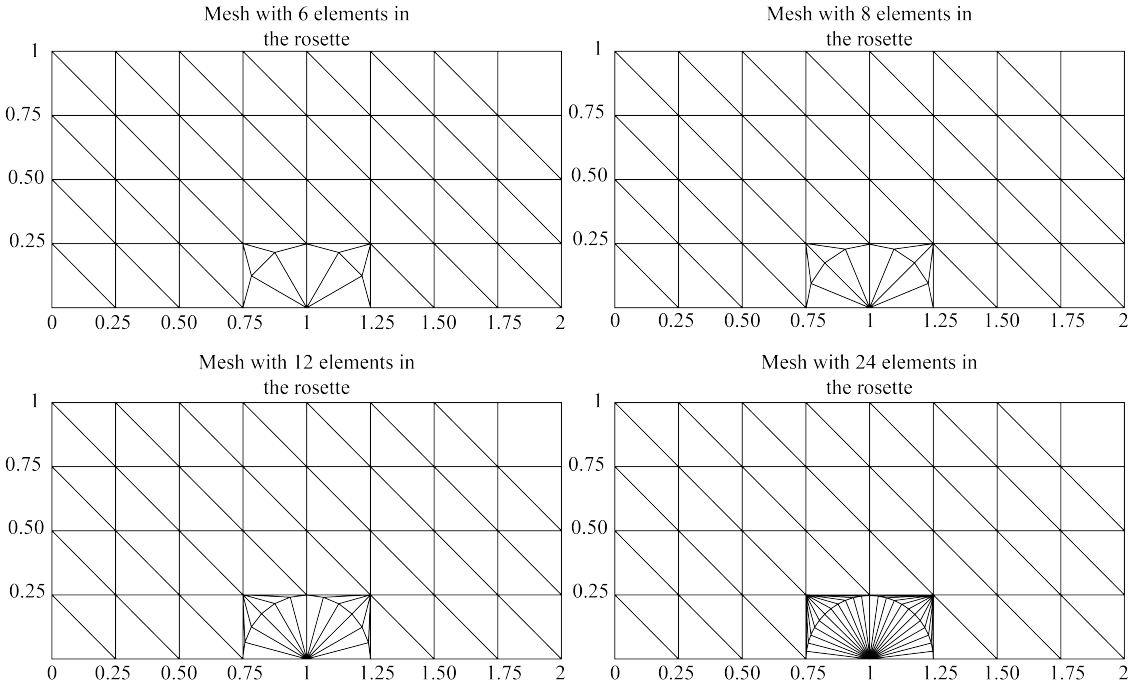


Figure 2.11 Meshes types employed up until now..

For 6 and 8 singular elements, rosette and transitioning elements are not specially deformed. However, as the number of singular elements grows, deformation becomes a problem.

2.9 Conditions of use

As we can find in several sources, (for instance [16], [11], [20]), a beneficial mesh trait is the use of equilateral-shaped elements, in our case equilateral triangles (or squares in other cases). Most elements are designed to bear a certain amount of distortion in order to fit more complex geometries, however an excessive deformation would dramatically affect the convergence of the solution and distort the final results of a mesh. Concepts like aspect ratio are used to calculate a mesh quality by certain softwares (among which we find Abaqus and ANSYS). Degenerate Elements augment the inaccuracy of FEA, and since our element has been obtained by

collapsing an equilateral element (hence introducing a bit of distortion in its design), and the test meshes did also stretch too much those singular elements and the following transition elements, changing this parameter could substantially affect our result. This is what Romero suggested as changing the mesh type, that is to say, finding a configuration that improves the element behaviour close to the crack tip and does not distort the solution close enough to the singular elements as to change its behavior.

Furthermore, some suggestions on element size and proportion appear in Abaqus handbooks and forums. It is recommended to keep the size of the non-singular elements around the quarter-point region as 0.7 times the singular-element size. The research in [19] remarks that quarter-point elements studies give a more accurate stress intensity factor than its quadrilateral counterpart, although specifies that natural isoparametric triangles work better than collapsed shapes. Moreover, remarks that its span angle should not exceed 60, thus rendering our already studied case of 2 singular elements in doubt, although reinforcing the idea of using a rosette configuration is still desireable. Such elements are suggested to occupy from 5 % to 25 % of the crack's length (which is later proven to be roughly the same for our element). Transition elements, those placed after the singular elements, are proposed to have a certain size in proportion to the quarter-point elements to correctly sense the singularity, being specially helpfull (according to some like [28]) when the proportion between the quarter-point is much smaller than the crack length.

Our main difference with the later studies is that we will be comparing the solution (K) not to an analytic solution of the problem but to another numeric solution, obtained with a fine mesh which has already converge (with the number of elements). As stated in the same paper, a 1 % accuracy would be more than reasonable. This explains the lack of error curves in the work, since we are comparing different shapes, not significantly augmenting the number of elements, so no shape is presumably better to compare.

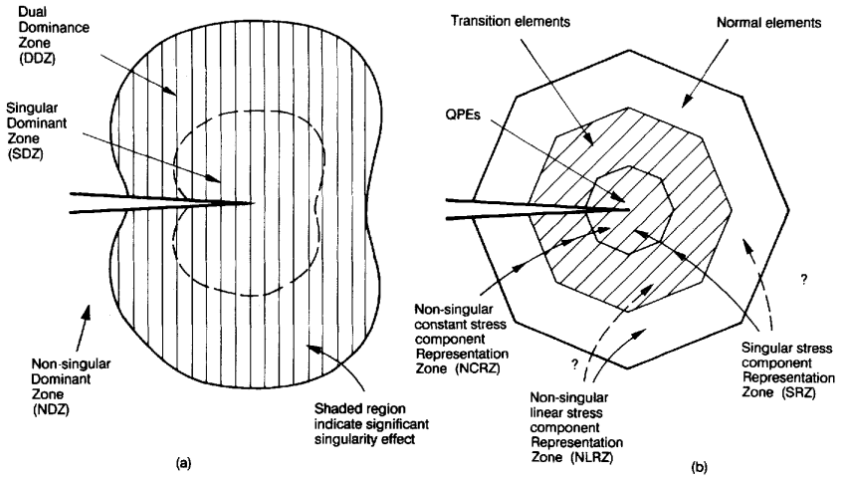


Figure 2.12 Taken from [19].

The same research also defined several zones: zones of dominance and zones of representation. The zones of dominance are: singular dominant zone (where stresses are dominated by the singularity, in the vicinity of the crack tip), non-singular dominant zone (where stresses are not relevantly affected by the singularity, far from the crack tip), and dual dominance zone (where neither component dominates, in-between the two others). The zones of representation are: singular stress component representation zone (the zone which we can model with singular elements), non-singular constant component stress component representation zone (the zone which can be modeled with non-singular, constant elements), and the non-singular linear stress component representation zone (which is understood to transition between both). Both zones of dominance and zones of representations should be balanced to obtain an optimal result. In our case, however, we will be centered in using constant elements instead of linear elements in the transition zone, supposing that linear behaviour will improve the solution and trying to achieve a good transition thanks to the element's shape and arrangement instead of linear shape function.

Ultimately, we will be studying the same regions but with a slightly different approach, and after the data is analysed and best-fitting data is chosen, we can estimate the real length of the zones with our final results (whose dimensions are summed in the last table) and the equations provided by Lim (variables described in notation): $L_{SRZ} = L_Q + \sum L_S = L_Q + \sum L_T + \sum L_N$.

3 First Parametric Study: effect of one and two crowns surrounding the singular elements

3.1 Parameter definitions

Let us begin by defining the parameters we will be modifying to determine what kind of meshing is optimal:

3.1.1 Singular element size

Let us call a_1 to the radius of the singular element: the distance between the crack tip and one of the vertices on the opposite side of the element (so that the whole element is encircled). For the sake of simplicity, let us call SER (Singular-Elements Region) to this region.

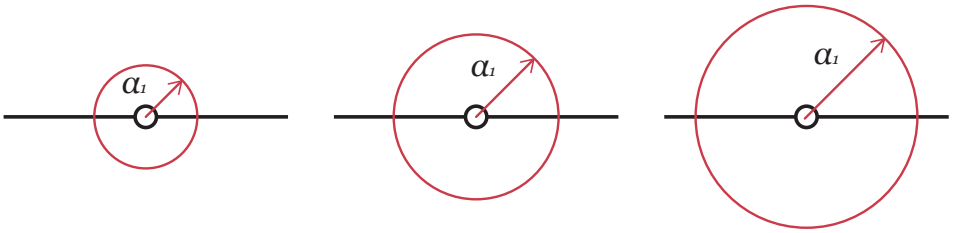


Figure 3.1 Different values of a_1.

For the case of 6 singular elements around the crack tip, with α_1 increasing towards the right:

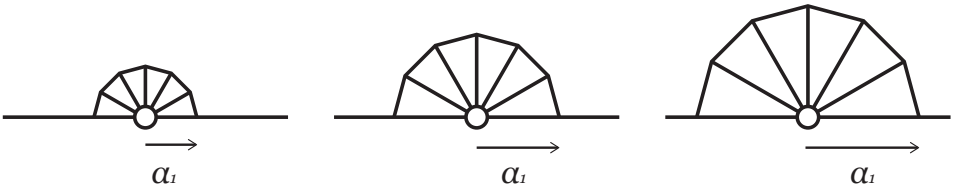


Figure 3.2 Different values of a_1 for six singular elements.

3.1.2 Number of singular elements

We will study the following cases: 4, 6, 8, 12 and 24 elements in a rosette pattern. Let us call n the number of elements in the neighborhood of the crack. We end up with these mesh options:

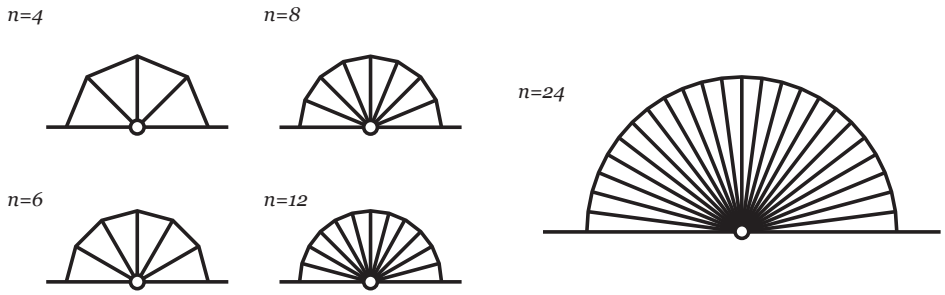


Figure 3.3 Different values of n for six singular elements.

3.1.3 Size of elements in the first crown

Let us assume, as stated in the previous knowledge section, that using one crown of elements (that is, a circular crown surrounding the singular elements) will be beneficial. By now, it will be meshed with non-singular elements. The two circumferences that define the crown are defined by radii a_1 and a_2. Thus a_1 is both the radius of the outer circumference of the SER and also of the inner circumference of the first crown. In the image, a_2 increases while a_1 remains constant, although they are independent from one another.

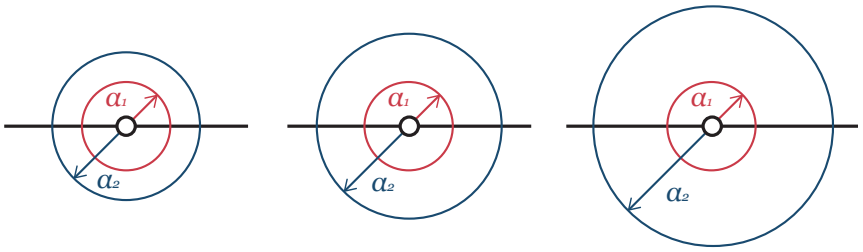


Figure 3.4 Different values of a_2 for six singular elements.

3.1.4 Type of the first crown

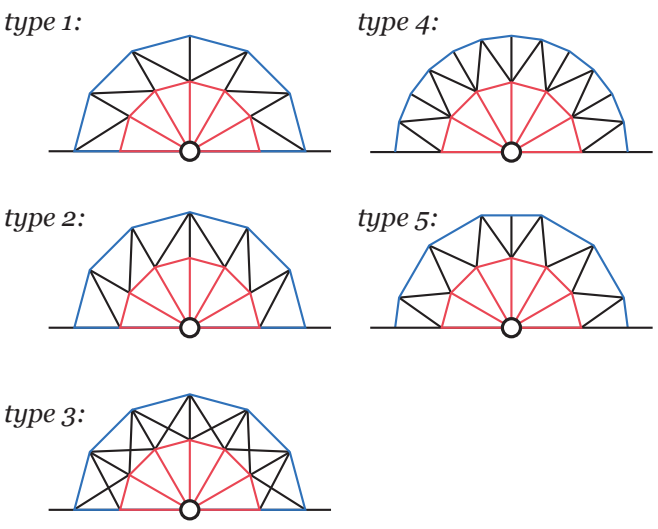


Figure 3.5 Different values of a_2 for six singular elements.

The different feasible shape of the elements in the first crown will be referred to as 'Type of the first crown'. We will study the following types of meshing, which are now introduced for $n = 6$ but will be generalized in the following section (it is easily extrapolatable to any n).

We can differentiate between radial crowns (types 1, 2, 3 and 4, in which the singular elements edges continue along the crown) and non-radial crowns (type 5, in which said edges do not continue). Also we can differentiate between simple crowns (types 1, 2 and 3, in which the number

of elements in the outer circumference is the same as in the inner one) and double crowns (types 4 and 5, in which the number of elements in the outer circumference is higher than those in the inner one).

3.1.5 Size of elements in the second crown

Akin to the circumference of radius a_2 used for the first crown, a third circumference with radius a_3 is used to define the second crown. The radius increases to the right keeping a_1 and a_2 constant. It is worth noting that the study can be conducted with or without this crown (as we will see later).

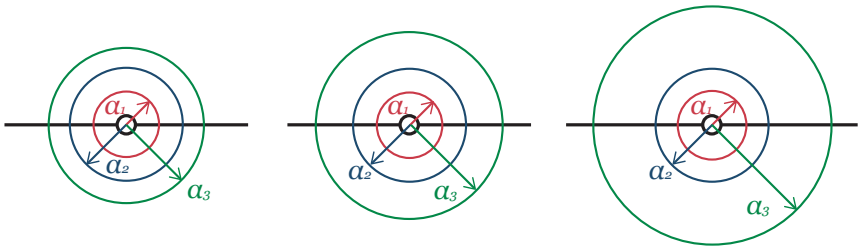


Figure 3.6 Increasing values of a_3.

3.1.6 Type of the second crown

Akin to the first crown, a new one with the same types can be introduced between a_2 and a_3. Let them be defined by the same codes (1, 2, 3, 4 and 5) meaning almost the same but with the following changes.

In the second crown's inner circumference there may be n elements (in case of first crown types 1, 2, 3), $2n$ elements (in case of the first crown type 4), or $n+2$ elements (in case of the first crown type 5). Hence, the 'type of the second crown'will have a slightly different shape depending on the first crown. Before further explanation in the following section, the changes are illustrated here.

In the appendix these diagrams are shown for the rest of case studies (4, 8, 12 and 24 elements).

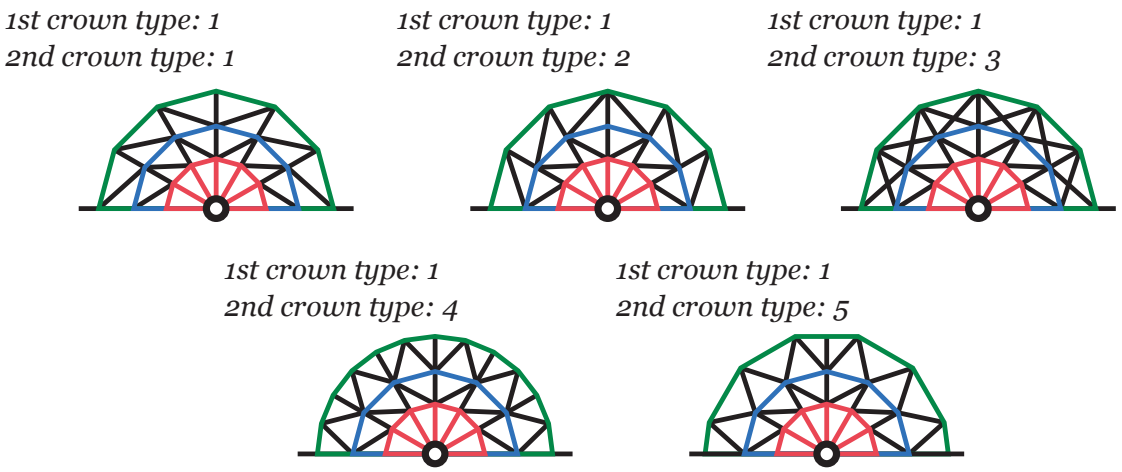


Figure 3.7 Second crown types for a type-1 first crown.

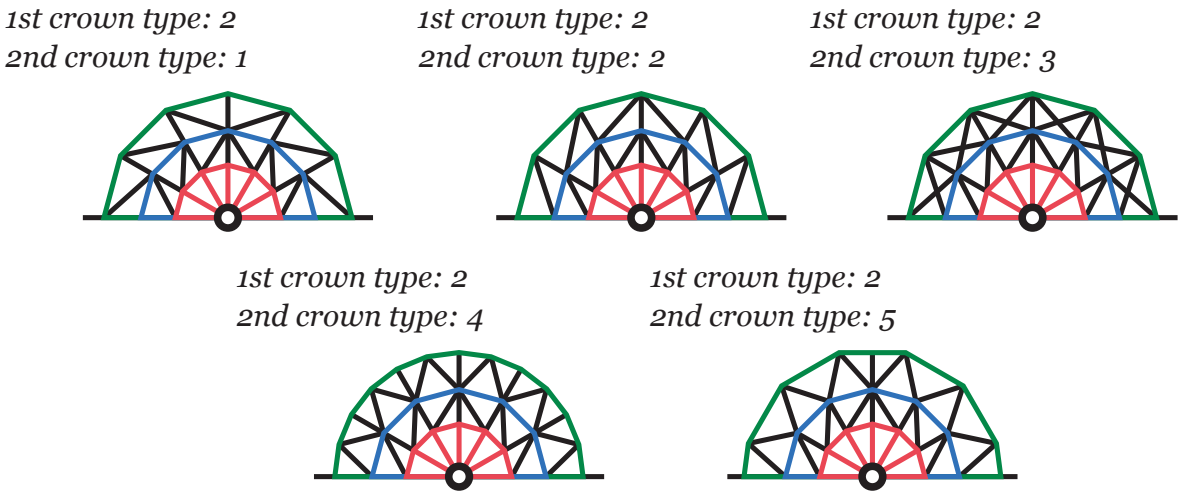


Figure 3.8 Second crown types for a type-2 first crown.

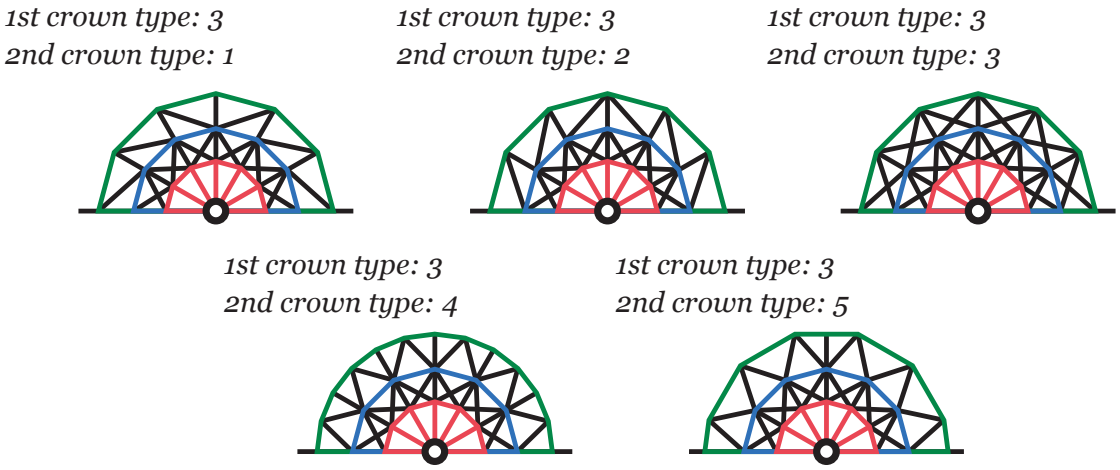


Figure 3.9 Second crown types for a type-3 first crown.

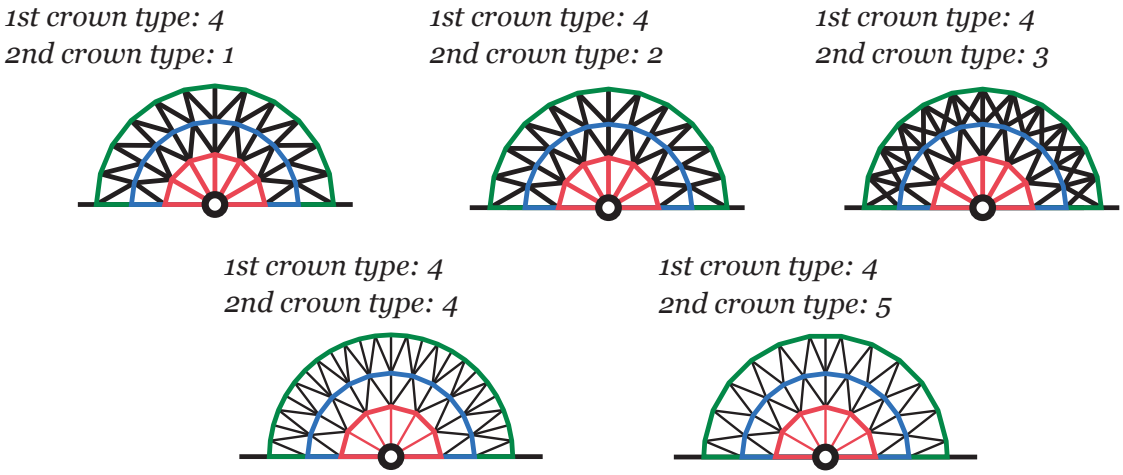


Figure 3.10 Second crown types for a type-4 first crown.

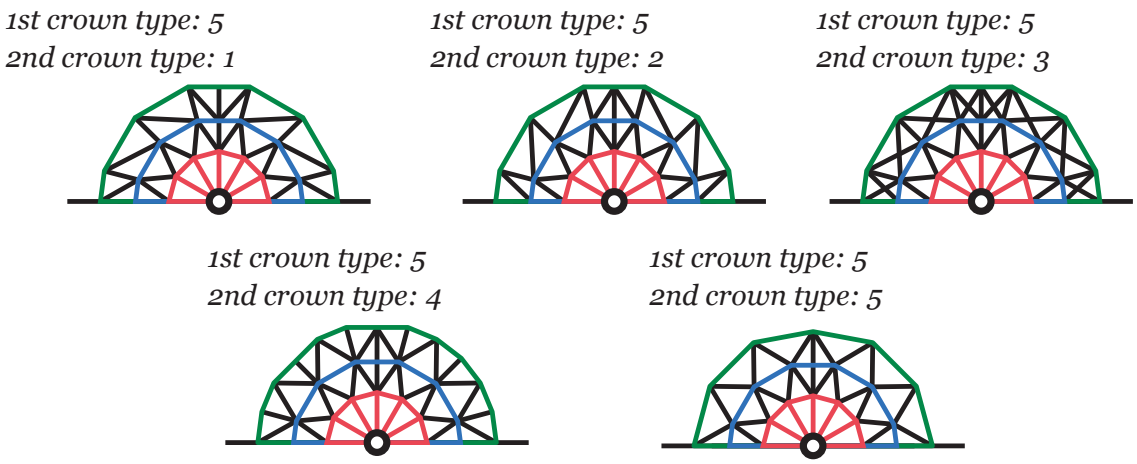


Figure 3.11 Second crown types for a type-5 first crown.

3.1.7 Rest of the mesh

What is left of the mesh should be kept as constant as possible in order to properly compare each case solution. If the mesh changed considerably, then the cause for a different solution would probably be the the fineness of the mesh and not our parameters. For this matter we will use a partition of the plate in the shape of a half-hexagon surrounding the last crown. This shape is employed since it results in a smoother transition between the square shape outside and the circular shapes inside (an heuristic proved adequate by the Department).

The exterior of the hexagon (EHR) is meshed with a constant number of mesh seeds, and its interior (IHR) is meshed freely. This last decision leads to consequences that will be further explained in the next section.

Additionally, to make the transition easier for the mesher, we will add two oblique cuts from the upper vertices of the square to the upper vertices of the half-hexagon, and a vertical cut halving the plate in two equal parts. Both will be meshed with an approximately constant number of seeds in each case.

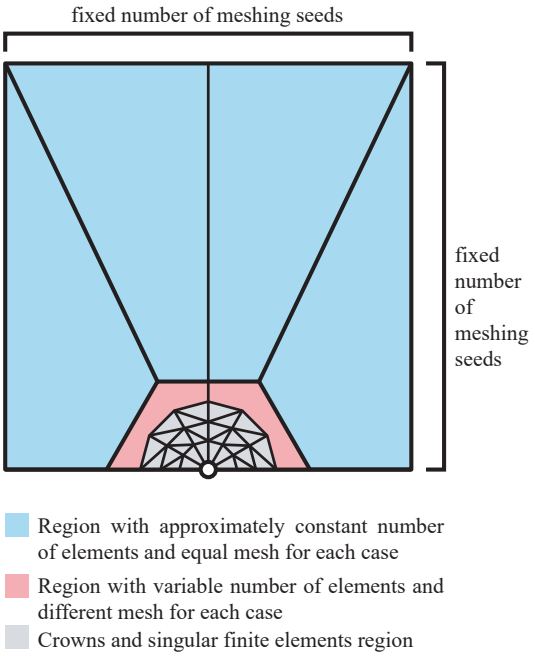


Figure 3.12 Whole model boundary conditions.

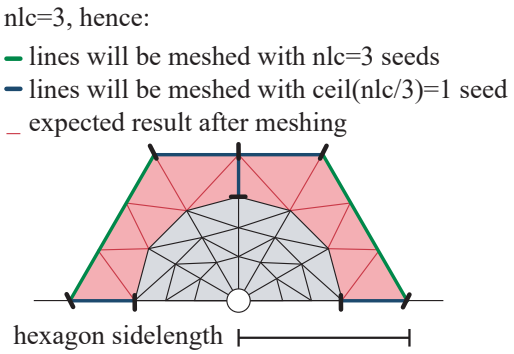


Figure 3.13 Half model boundary conditions.

After several preliminary tests, we will try two different options for the hexagon meshing:

Option a Using elements the same size as the rest of the mesh.

Option b Using a concrete number of elements depending on the number of elements in the outer circumference of the last crown. Thus we will ensure a smoother transition between hexagon and circumference. The seed options will be 'finer'(so that the mesher algorithm will increase its number if needed). The values are the following:

For the oblique hexagon lines the same number of seeds as elements on the outer crown (which we will call nlc).

For the horizontal and vertical lines from the half-hexagon to the crown (counting both sides of the upper horizontal line independently) will be meshed with ceil(nlc/ 3) seeds.

See the image for the case $nlc = 3$.

Finally, since the singular elements and crowns must always be inside the half-hexagon, it will be easier to express the radii (a_i) in a non-dimensional form by dividing it by the hexagon side-length (instead of using the dimensional measure). Hence, let us define the parameters a_i as:

$$\alpha_i = \frac{a_i}{hexagonsidelength} \; with \; i = 1, 2, 3 \tag{3.1}$$

Which we will be using from now on.

3.1.8 Nomenclature

To identify cases we will use the following naming code:

- Cases with only one crown:
$$nE\_tipo\_tc1\_tipo\_0\_\alpha_1\_\alpha_2 \tag{3.2}$$
- Cases with two crowns:
$$nE\_tipo\_tc1\_tipo\_tc2\_\alpha_1\_\alpha_2 \tag{3.3}$$

Where:

- n: number of singular elements
- tc1: type of the first crown
- tc2: type of the second crown
- α_1: radius of the circle surrounding the finite singular elements divided by the half-hexagon side length.
- α_2: radius of the circle surrounding the first crown finite elements divided by the half-hexagon side length.
- α_3: radius of the circle surrounding the second crown finite elements divided by the half-hexagon side length.

3.2 Cut definition

We need to find a way to mathematically express the different types of crown previously defined. Abaqus defines a straight line segment in a sketch as:

```
s.Line(point1=( x_initial_point , y_initial_point ), point2=( x_final_point , y_final_point ))
```

Script 3.1 Python script for cut-line definition.

In which sometimes the sketch is done over s1 instead of s.

Our strategy then will be generating several character strings, each line representing a desired segment which has to coincide with the edge of the elements in the crown. The strings can be achieved by using loops for each half of the crown, since each inner-circumference node connects in the same manner with the outer-circumference node as the following one.

The only step left is naming the nodes and finding their coordinates so that they can be connected by segments. Let us enumerate them counter clockwise for each circumference. With that in mind, and the crown diagrams from the previous section, we can identify the initial and final points of each segment. For type-5, where there are n + 2 nodes, we will use a 2n-polygon since the first and last segments twice as short as the other segments.

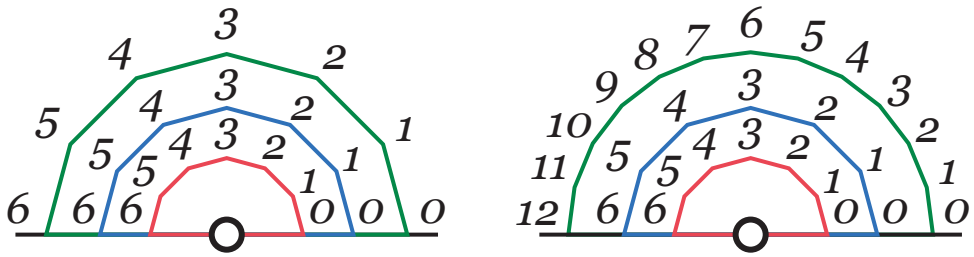


Figure 3.14 Node enumeration in crowns.

These nodes are placed on the vertices of regular n-polygons inscribed in circumferences with known radii a_1, a_2 and a_3 (in which n is the number of elements on the circumference, that depends on the number of singular elements and the type of the following crowns). Hence, their coordinates can be easily calculated. They will be introduced in the x1, x2, x3 and y1, y2, y3 vectors (x or y defining the axis and 1, 2 and 3 defining the circumference inside out). Note that the vectors size will not generally coincide.

```
N   =   2*nofelements
n   =   list(range(0, nofelements+1))
n2  =   list(range(0, nofelements+1))
n3  =   list(range(0, nofelements+1))
rci =   sidelength*alpha1
rco =   sidelength*alpha2
rco2=   sidelength*alpha3
x1  =   list(range(0, nofelements+1))
y1  =   list(range(0, nofelements+1))
x2  =   list(range(0, nofelements+1))
y2  =   list(range(0, nofelements+1))
x3  =   list(range(0, nofelements+1))
y3  =   list(range(0, nofelements+1))
```

Script 3.2 Python script to define crown vertices.

The segments are arranged as follows (this is the mathematical definition of the 'Type'):

```
%% Variables initialization
clear; clc;
code = [];
xvect= [];
yvect= [];
xvect = [];

%% Modifiable parameters
mark          = 's'; % Change by 's1' depending on the
    sketch
nofelements = 6; %% Number of elements in the interior
     circumference
xi            = 'x1'; % Choose which is the interior
    circumference (x1 if it is the first crown, x2
    if it is the second crown)
xj            = 'x2'; % Choose which is the exterior
    circumference (x2 if it is the first crown, x3
    if it is the second crown)
x\_ct         = 0;  % crack tip x coordinate
y\_ct         = 0;  % crack tip y coordinate
tipo          = 4;  % Type of crown

%% Creation of a matrix with the shape [x\_initial1 x\
    _final1; x\_initial2 x\_final2; ...] with a
    segment in each row.

%% Type 1
if tipo == 1,
    for s1=0:nofelements/2
        s=num2str(s1);
        sM1=num2str(s1+1);
```

```
            sm1=num2str(s1-1);
            Ds=num2str(2*s1);
            line=[xj,'[',s,']  ',xi,'[',sM1,']'];
            line2=[xi,'[',sM1,']  ',xj,'[',sM1,']'];
            xvect=strvcat(xvect,line,line2);
        end
        xvect(end-1:end,:)=[];
        for s1=nofelements/2:nofelements
            s=num2str(s1);
            sM1=num2str(s1+1);
            sm1=num2str(s1-1);
            Ds=num2str(2*s1);
            line=[xi,'[',s,']  ',xj,'[',sM1,']'];
            line2=[xj,'[',sM1,']  ',xi,'[',sM1,']'];
            xvect=strvcat(xvect,line,line2);
        end
        xvect(end-2:end,:) = [];
        xcorte = xvect;
end

%% Type 2
if tipo == 2,
        for s1=0:nofelements/2
            s=num2str(s1);
            sM1=num2str(s1+1);
            sm1=num2str(s1-1);
            Ds=num2str(2*s1);
            line=[xi,'[',s,']  ',xj,'[',sM1,']'];
            line2=[xi,'[',sM1,']  ',xj,'[',sM1,']'];
            xvect=strvcat(xvect,line,line2);
        end
        xvect(end-1:end,:)=[];
        for s1=nofelements/2:nofelements
            s=num2str(s1);
            sM1=num2str(s1+1);
            sm1=num2str(s1-1);
            Ds=num2str(2*s1);
            line=[xj,'[',s,']  ',xi,'[',sM1,']'];
            line2=[xj,'[',sM1,']  ',xi,'[',sM1,']'];
            xvect=strvcat(xvect,line,line2);
        end
        xvect(end-2:end,:)=[];
        xcorte=xvect;
end

%% Type 3
if tipo == 3,
        disp('add lines obtained with type 1 plus the ones
            obtained with type 2')
end

%% Type 4
if tipo == 4,
        for s1=0:nofelements-1
            s=num2str(s1);
            sM1=num2str(s1+1);
            sm1=num2str(s1-1);
            Ds=num2str(2*s1);
            Ds_M1=num2str(2*s1+1);
            DsM1=num2str(2*(s1+1));
            line=[xi,'[',s,']  ',xj,'[',Ds_M1,']'];
            line2=[xj,'[',Ds_M1,']  ',xi,'[',sM1,']'];
            line3=[xi,'[',sM1,']  ',xj,'[',DsM1,']'];
            xvect=strvcat(xvect,line,line2,line3);
        end
        xvect(end,:)=[];
        xcorte=xvect;
end

%% Type 5
if tipo == 5,
        for s1=0:nofelements-1
            s=num2str(s1);
            sM1=num2str(s1+1);
            sm1=num2str(s1-1);
            Ds=num2str(2*s1);
            Ds_M1=num2str(2*s1+1);
            DsM1=num2str(2*(s1+1));
            line=[xi,'[',s,']  ',xj,'[',Ds_M1,']'];
            line2=[xj,'[',Ds_M1,']  ',xi,'[',sM1,']'];
            xvect=strvcat(xvect,line,line2);
        end
        xcorte=xvect;
end

%% finding separators
for s1 = [1:size(xvect,1)]
        separador(s1,:) = strfind(xvect(s1,:),'  ');
end

%% Creation of a matrix with the shape [y\_initial1  y\
    _final1; y\_initial2  y\_final2; ...] with a
    segment in each row, from the preexisting matrix
    .
for s1 = [1:size(xvect,1)]
        yvect(s1,:) = strrep(xvect(s1,:),'x','y');
end

%% Creation of the code lines
for s1 = [1:size(xvect,1)]
        codeline = [mark,'.Line(point1=(',num2str(x_ct),'+
            ',(xvect(s1,1:separador(s1,1)-1)),',',num2str(
            y_ct),'+',(yvect(s1,1:separador(s1,1)-1)),'),
            point2=(',num2str(x_ct),'+',(xvect(s1,separador(
            s1,1)+1:end)),',',num2str(y_ct),'+',yvect(s1,
            separador(s1,1)+1:end),'))'];
        code=[code;codeline];
end

%% Display (copy and paste in the Python code)
code
```

Script 3.3 MATLAB Script to define cuts as vectors.

3.3 Mesh generation

After defining each and every parameter, we need to generate a mesh and to control its parameters. In fact, it is precise to generate a high amount of meshes due to all the possible parameter combinations. After that, they should be stored in a file and simulated with the same boundary conditions for every case. By comparing the results of our study to a reference result (which we consider to be correct up to the desired amount of digits) we can establish the optimal relationship between parameters (as a function of all the parameters involved).

We will take advantage of the perks of Abaqus'meshing algorithm although they can also be generated in MATLAB. Since our main purpose is implementing an Abaqus routine, it may be preferable to use said commercial code, which we presume to be of higher-quality results than a homemade code. There is also a possibility to export the Abaqus-generated meshes in an easily manageable '.inp'file which can be read in MATLAB.
The process to make a single mesh is as follows. The commands employed by Abaqus is added after each step, specified for the case 6E_tipo_1_tipo_1. Abaqus allows us to record each and every command in a '.py'file by clicking on: File → Macro Manager → Create. The resulting archive is editable so that it can be modified to easily implement changes in parameters. The lines that are affected by our parameters are highlighted in yellow. are highlighted in yellow, and the lines that define the regions that will be cut are highlighted in red. Information presented in [29].

- **Step 01: Create the part** Creation of a solid planar deformable 2D Shell, sketch as a square from (− 0.5,0.5) to (0.5,0.5). Part module → Create Part → 2D Planar → Deformable, Solid, Approximate size: 2. Rectangle coordinates highlighted.

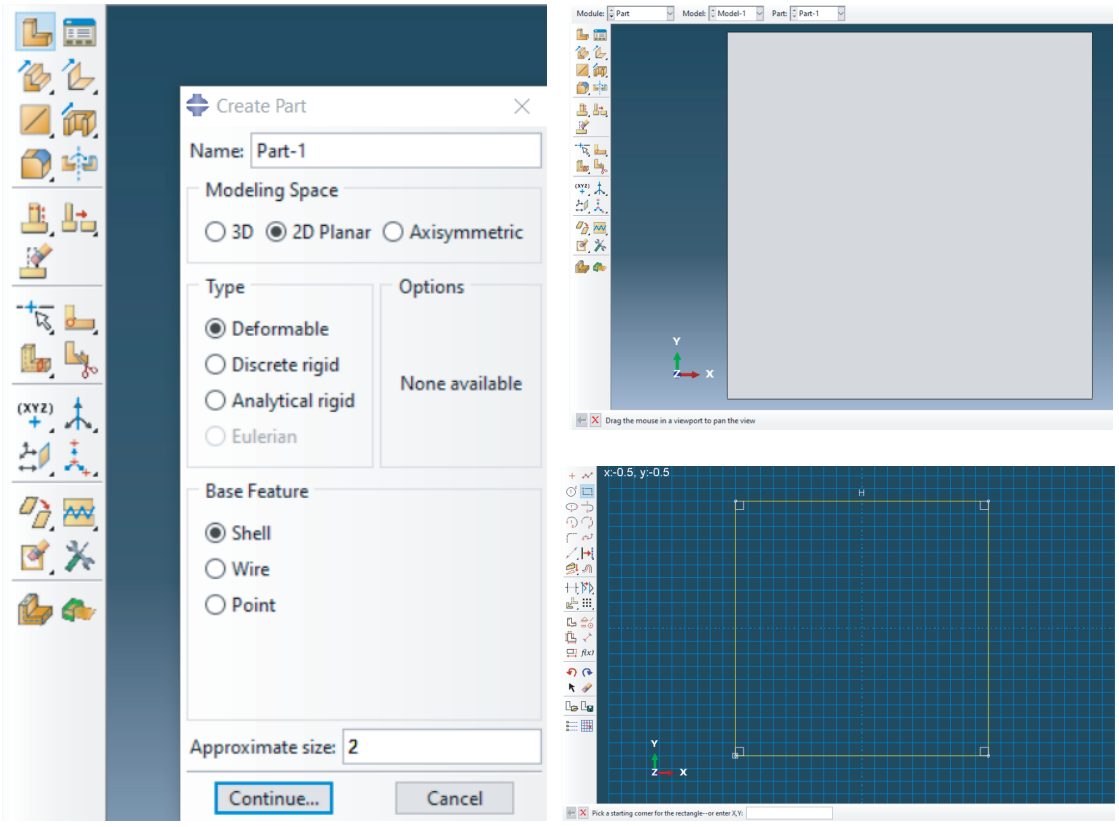


Figure 3.15 Part features.

```
\#____________________\# Step 01
Mdb()
\colorbox{yellow}{texto  resaltado }
s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=2.0)
g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
s.setPrimaryObject(option=STANDALONE)
s.rectangle(point1=(-0.5, 0.0), point2=(0.5, 1.0))
p = mdb.models['Model-1'].Part(name='Part-1', dimensionality=TWO_D_PLANAR,
    type=DEFORMABLE_BODY)
p = mdb.models['Model-1'].parts['Part-1']
p.BaseShell(sketch=s)
```

```
s.unsetPrimaryObject()
p = mdb.models['Model-1'].parts['Part-1']
session.viewports['Viewport: 1'].setValues(displayedObject=p)
del mdb.models['Model-1'].sketches['__profile__']
```

Script 3.4 Step 01 in Python.

- **Step 02: Make the Half-Hexagon** Cutting of the half-hexagon (from coordinates created assuming Origin of Coordinates on the crack tip, else we must add the crack tip coordinates plus the half-hexagon ones), the auxiliary lines from the upper vertices of the plates to the upper vertices of the half-hexagon, and the vertical symmetry line. Part module → Partition Face: Sketch → Select face, then draw lines. In yellow, the lines sketched joining auxiliary points.

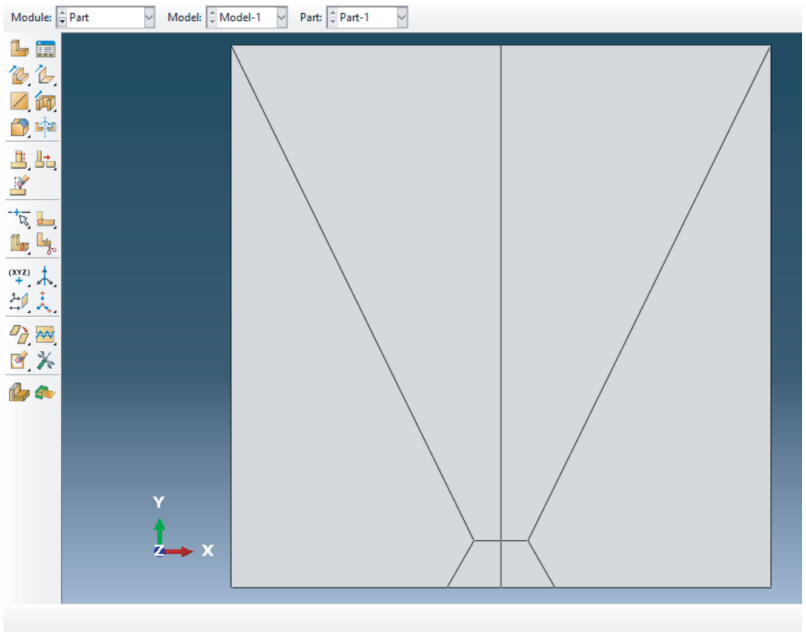


Figure 3.16 Result of the Step 02.

```
p = mdb.models['Model-1'].parts['Part-1']
f1, e1, d2 = p.faces, p.edges, p.datums
t = p.MakeSketchTransform(sketchPlane=f1[0], sketchPlaneSide=SIDE1, origin=(
    0.0, 0.5, 0.0))
s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=2.82,
    gridSpacing=0.07, transform=t)
g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
s.setPrimaryObject(option=SUPERIMPOSE)
p = mdb.models['Model-1'].parts['Part-1']
p.projectReferencesOntoSketch(sketch=s, filter=COPLANAR_EDGES)
s.Line(point1=(0.0, -0.51), point2=(0.0+xpaux[0], -0.5+ypaux[0]))
s.Line(point1=(0.0+xpaux[0], -0.5+ypaux[0]), point2=(0.0+xpaux[1], -0.5+ypaux[1]))
s.Line(point1=(0.0+xpaux[1], -0.5+ypaux[1]), point2=(0.0+xpaux[2], -0.5+ypaux[2]))
s.Line(point1=(0.0+xpaux[2], -0.5+ypaux[2]), point2=(0.0+xpaux[3], -0.5+ypaux[3]))
s.Line(point1=(0.0+xpaux[3], -0.5+ypaux[3]), point2=(0.0+xpaux[4], -0.5+ypaux[4]))
s.Line(point1=(0.0+xpaux[4], -0.5+ypaux[4]), point2=(0.0, -0.51))
s.Line(point1=(0.5, 0.5), point2=(0.0+xpaux[1], -0.5+ypaux[1]))
s.Line(point1=(-0.5, 0.5), point2=(0.0+xpaux[3], -0.5+ypaux[3]))
s.Line(point1=(0.0, -0.5), point2=(0.0, 0.5))
p = mdb.models['Model-1'].parts['Part-1']
f = p.faces
pickedFaces = f.getSequenceFromMask(mask=('[#1 ]', ), )
e, d1 = p.edges, p.datums
```

```
p.PartitionFaceBySketch(faces=pickedFaces, sketch=s)
s.unsetPrimaryObject()
del mdb.models['Model-1'].sketches['__profile__']
```

Script 3.5 Step 02 in Python.

- **Step 03: Make Circumferences** Cutting of the two circumferences centered in the crack tip. The inside region corresponds to singular elements, and the in-between region to the first crown. If we were to add a second crown, a new exterior circumference should be cut. Part module → Partition Face: Sketch → Select face (inside half hexagon), then draw two/three circumferences centered on the midpoint of the shell's lower edge and with radii α_1, α_2, α_3 times the hexagon side-length.. Do not use the third command when there is only one crown.

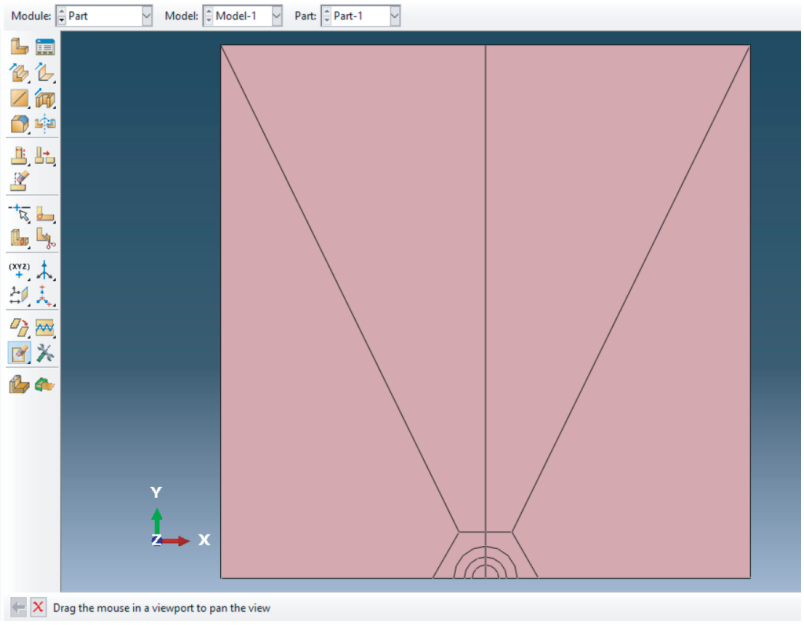


Figure 3.17 Result of Step 03.

```
p = mdb.models['Model-1'].parts['Part-1']
f, e1, d2 = p.faces, p.edges, p.datums
t = p.MakeSketchTransform(sketchPlane=f[1], sketchPlaneSide=SIDE1, origin=(
    0.097222, 0.096225, 0.0))
s1 = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
    sheetSize=1.11, gridSpacing=0.02, transform=t)
g, v, d, c = s1.geometry, s1.vertices, s1.dimensions, s1.constraints
s1.setPrimaryObject(option=SUPERIMPOSE)
p = mdb.models['Model-1'].parts['Part-1']
p.projectReferencesOntoSketch(sketch=s1, filter=COPLANAR_EDGES)
s1.CircleByCenterPerimeter(center=(x_ct, y_ct), point1=(x_ct+rci, y_ct))
s1.CircleByCenterPerimeter(center=(x_ct, y_ct), point1=(x_ct+rco, y_ct))
s1.CircleByCenterPerimeter(center=(x_ct, y_ct), point1=(x_ct+rco2, y_ct))
p = mdb.models['Model-1'].parts['Part-1']
f = p.faces
pickedFaces = f.getSequenceFromMask(mask=('[#a ]', ), )
e, d1 = p.edges, p.datums
p.PartitionFaceBySketch(faces=pickedFaces, sketch=s1)
s1.unsetPrimaryObject()
del mdb.models['Model-1'].sketches['__profile__']
```

Script 3.6 Step 03 in Python.

- **Step 04: Remove the SER.** Removal of the inner circle, which will be substituted by a distribution of singular elements later, when the case is simulated in MATLAB. Part Module → Remove Faces, then select both halves of the innermost circle.

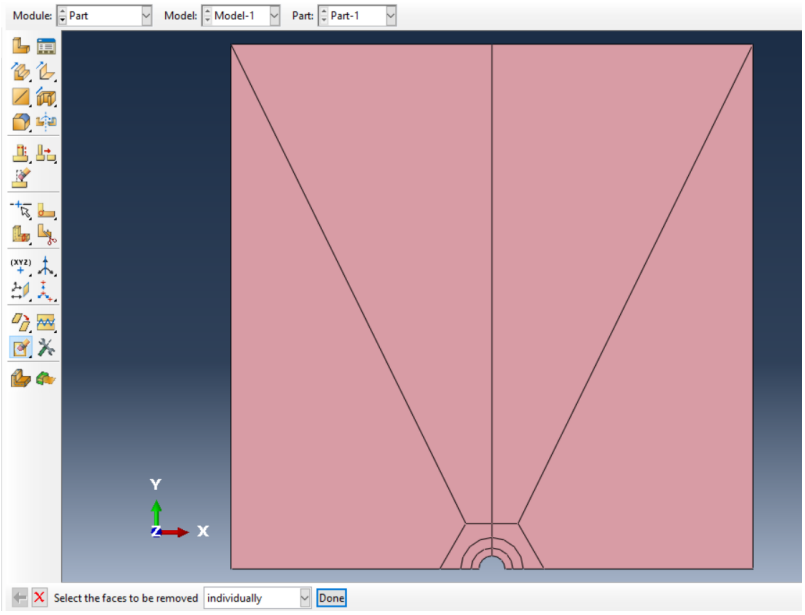


Figure 3.18 Result of Step 04.

```
session.viewports['Viewport: 1'].view.setValues(nearPlane=2.63361,
    farPlane=3.02325, width=1.10348, height=0.554118, cameraPosition=(
    0.0496831, 0.207573, 2.82843), cameraTarget=(0.0496831, 0.207573, 0))
p = mdb.models['Model-1'].parts['Part-1']
f1 = p.faces
p.RemoveFaces(faceList = f1[5:6]+f1[7:8], deleteCells=False)
```

Script 3.7 Step 04 in Python.

- **Step 05: Cut the first crown.** Cutting of the first crown shape according to its type. Also, cutting of the second crown shape if necessary. The figure cutting uses straight lines to replicate the edges of the elements as imposed by the crown type. This can be easily achieved since the edges of said elements coincide with segments that connect the vertices of regular polygons inscribed inside the pre-existing circumferences. Hence, we can just calculate the vertices coordinates and select them as the starting/ending points of segments (according to the desired crown type). Part Module → Select faces between remaining circumferences, then draw segments from coordinates.

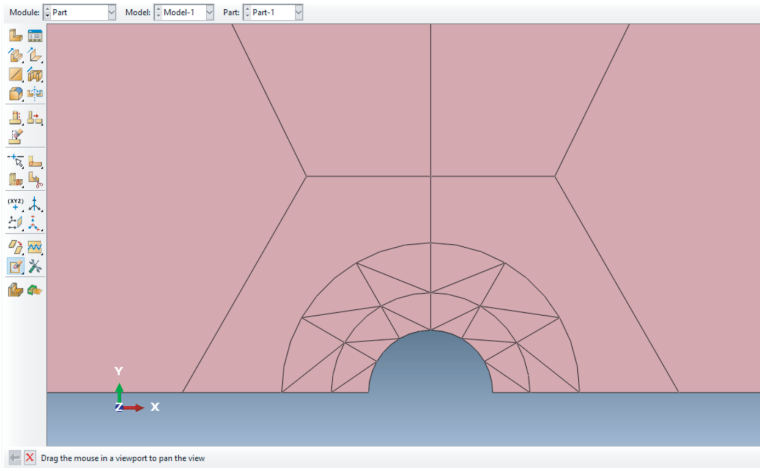


Figure 3.19 Close-up of the resulting crown.

```
p = mdb.models['Model-1'].parts['Part-1']
f, e1, d2 = p.faces, p.edges, p.datums
t = p.MakeSketchTransform(sketchPlane=f[1], sketchPlaneSide=SIDE1, origin=(
    y_ct, 0.082041, 0.0))
s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=0.91,
    gridSpacing=0.02, transform=t)
g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
s.setPrimaryObject(option=SUPERIMPOSE)
p = mdb.models['Model-1'].parts['Part-1']
p.projectReferencesOntoSketch(sketch=s, filter=COPLANAR_EDGES)

# First crown
s.Line(point1=(x_ct+x2[0] ,y_ct+y2[0] ), point2=(x_ct+ x1[1],y_ct+ y1[1]))
s.Line(point1=(x_ct+x1[1] ,y_ct+y1[1] ), point2=(x_ct+ x2[1],y_ct+ y2[1]))
s.Line(point1=(x_ct+x2[1] ,y_ct+y2[1] ), point2=(x_ct+ x1[2],y_ct+ y1[2]))
s.Line(point1=(x_ct+x1[2] ,y_ct+y1[2] ), point2=(x_ct+ x2[2],y_ct+ y2[2]))
s.Line(point1=(x_ct+x2[2] ,y_ct+y2[2] ), point2=(x_ct+ x1[3],y_ct+ y1[3]))
s.Line(point1=(x_ct+x1[3] ,y_ct+y1[3] ), point2=(x_ct+ x2[3],y_ct+ y2[3]))
s.Line(point1=(x_ct+x1[3] ,y_ct+y1[3] ), point2=(x_ct+ x2[4],y_ct+ y2[4]))
s.Line(point1=(x_ct+x2[4] ,y_ct+y2[4] ), point2=(x_ct+ x1[4],y_ct+ y1[4]))
s.Line(point1=(x_ct+x1[4] ,y_ct+y1[4] ), point2=(x_ct+ x2[5],y_ct+ y2[5]))
s.Line(point1=(x_ct+x2[5] ,y_ct+y2[5] ), point2=(x_ct+ x1[5],y_ct+ y1[5]))
s.Line(point1=(x_ct+x1[5] ,y_ct+y1[5] ), point2=(x_ct+ x2[6],y_ct+ y2[6]))
# Second crown
s1.Line(point1=(x_ct+x3[0] ,y_ct+y3[0] ), point2=(x_ct+x2[1],y_ct+y2[1]))
s1.Line(point1=(x_ct+x2[1] ,y_ct+y2[1] ), point2=(x_ct+x3[1],y_ct+y3[1]))
s1.Line(point1=(x_ct+x3[1] ,y_ct+y3[1] ), point2=(x_ct+x2[2],y_ct+y2[2]))
s1.Line(point1=(x_ct+x2[2] ,y_ct+y2[2] ), point2=(x_ct+x3[2],y_ct+y3[2]))
s1.Line(point1=(x_ct+x3[2] ,y_ct+y3[2] ), point2=(x_ct+x2[3],y_ct+y2[3]))
s1.Line(point1=(x_ct+x2[3] ,y_ct+y2[3] ), point2=(x_ct+x3[3],y_ct+y3[3]))
s1.Line(point1=(x_ct+x2[3] ,y_ct+y2[3] ), point2=(x_ct+x3[4],y_ct+y3[4]))
s1.Line(point1=(x_ct+x3[4] ,y_ct+y3[4] ), point2=(x_ct+x2[4],y_ct+y2[4]))
s1.Line(point1=(x_ct+x2[4] ,y_ct+y2[4] ), point2=(x_ct+x3[5],y_ct+y3[5]))
s1.Line(point1=(x_ct+x3[5] ,y_ct+y3[5] ), point2=(x_ct+x2[5],y_ct+y2[5]))
s1.Line(point1=(x_ct+x2[5] ,y_ct+y2[5] ), point2=(x_ct+x3[6],y_ct+ y3[6]))
p = mdb.models['Model-1'].parts['Part-1']
f = p.faces
pickedFaces = f.getSequenceFromMask(mask=('[#a ]', ), )
e, d1 = p.edges, p.datums
p.PartitionFaceBySketch(faces=pickedFaces, sketch=s)
s.unsetPrimaryObject()
del mdb.models['Model-1'].sketches['\_\ _profile \_\_']
```

Script 3.8 Step 05 in Python.

- **Step 06: Mesh seeds.** Mesh seeds are placed on the crown segments in order to make each sub-part coincide with a triangular element. To do so, each segment must coincide with an edge, without the possibility to let its number change. Hence, we will meshed with 'tri free'conditions. Mesh module → Seed Edges → Choose the crown edges → Done → Basic: (By number: Number of Elements: 1; Bians: None); Constraints: 'Do not allow the number of elements to change'.

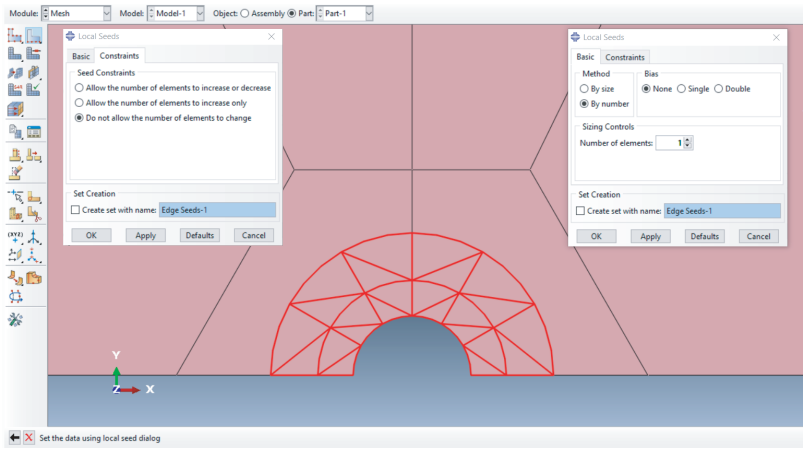


Figure 3.20 Mesh seeds of the crowns segments 06.

```
p = mdb.models['Model-1'].parts['Part-1']
e = p.edges
pickedEdges = e.getSequenceFromMask(mask=('[#e1fffff #ffff ]', ), )
p.seedEdgeByNumber(edges=pickedEdges, number=1, constraint=FIXED)
```

Script 3.9 Step 06 in Python.

- **Step 07: Mesh seeds.** Mesh seeds for b cases are placed on the oblique edges of the half-hexagon, with the values explained in the previous section. Mesh → Seed Edges → Choose the oblique half-hexagon edges → Done → Basic: (By number: Number of Elements: number of elements on the outer circumference; Bias: None); Constraints: Álow the number of elements to increase or decrease'.

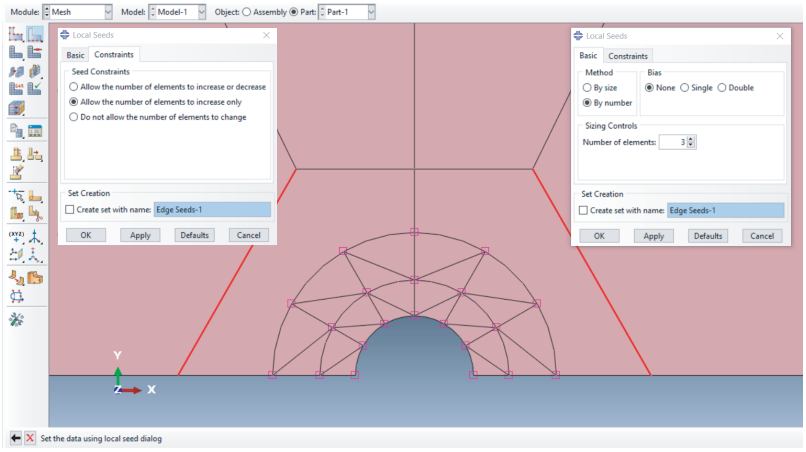


Figure 3.21 Mesh seeds of the half-hexagon oblique edges.

```
p = mdb.models['Model-1'].parts['Part-1']
e = p.edges
pickedEdges = e.getSequenceFromMask(mask=('[#8000000 #20000 ]', ), )
p.seedEdgeByNumber(edges=pickedEdges, number= number of elements in the last crown)
```

Script 3.10 Step 07 in Python.

- **Step 08: Mesh seeds.** Remaining mesh seeds for b cases are placed on the horizontal edges of the half-hexagon and the vertical edge inside ofit, with the values explained in the previous section. Mesh → Seed Edges → Choose the two horizontal hexagon edges and the three segments between the half hexagon and the outer crown → Done → Basic: (By number: Number of Elements: ceiling of the result of the division of the number of elements on the outer circumference divided by three; Bias: None); Constraints: “Allow the number of elements to increase or decrease”.

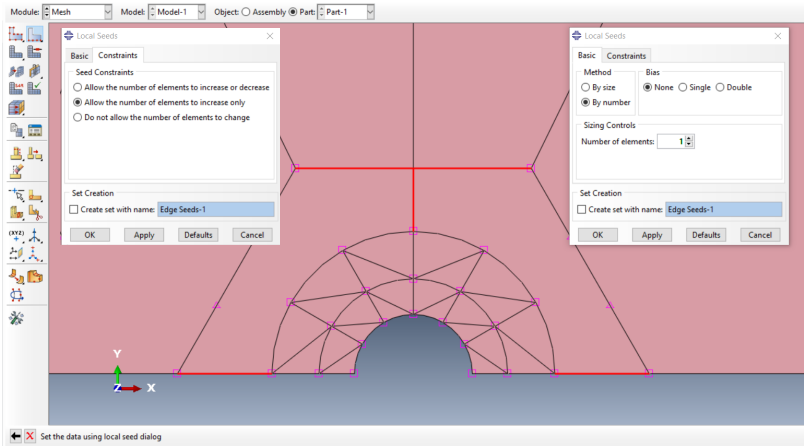


Figure 3.22 Mesh seeds for the remaining segments..

```
p = mdb.models['Model-1'].parts['Part-1']
e = p.edges
pickedEdges = e.getSequenceFromMask(mask=('[16000000 50000 ]', ), )
p.seedEdgeByNumber(edges=pickedEdges, number= ceil(number of elements in the last crown / 3))
```

Script 3.11 Step 08 in Python.

- **Step 09: Mesh seeds.** Mesh seeds. Let us impose an approximate velue of global seeds in order not to let the size of the crown elements determine the average size of the elements. Out of the hexagon and far away from it, everything should remain approximately equal for every study. Mesh module → Seed Part → (Approximate global size: 0.05, tick on curvature control: Maximum deviation factor: 0.1, Minimum size control: by fraction of global size: 0.1).

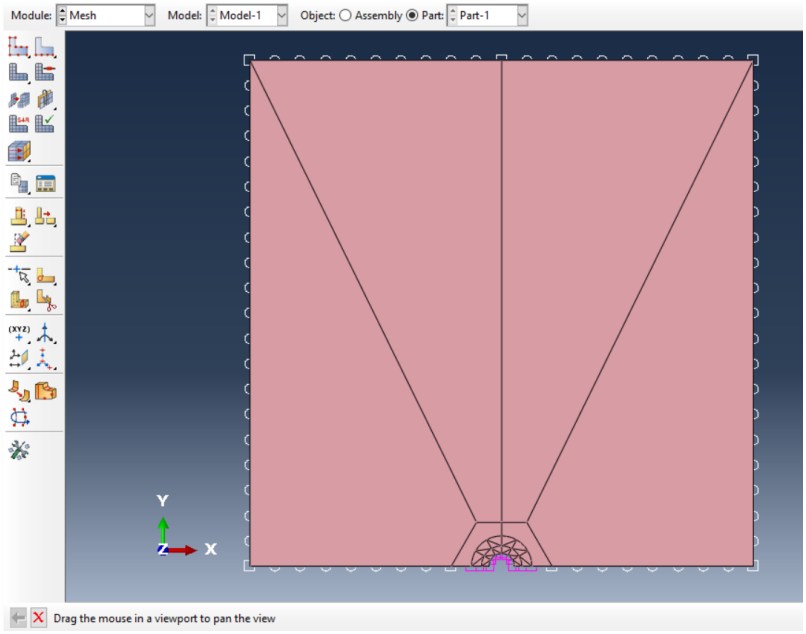


Figure 3.23 Global seeds.

```
p = mdb.models['Model-1'].parts['Part-1']
p.seedPart(size=0.05, deviationFactor=0.1, minSizeFactor=0.1)
```

Script 3.12 Step 09 in Python.

- **Step 10: Mesh control.** After several tries, a free triangular mesh confers symmetry to the model outside the half-hexagon. The structured meshes also produces a symmetric outcome, but triangles become further from equilateral as we distance from the half-hexagon. That is not a positive trait for a mesh. Mesh Module → Assign Mesh Controls → Select the whole part → (Element Shape: Tri, Technique: Free).

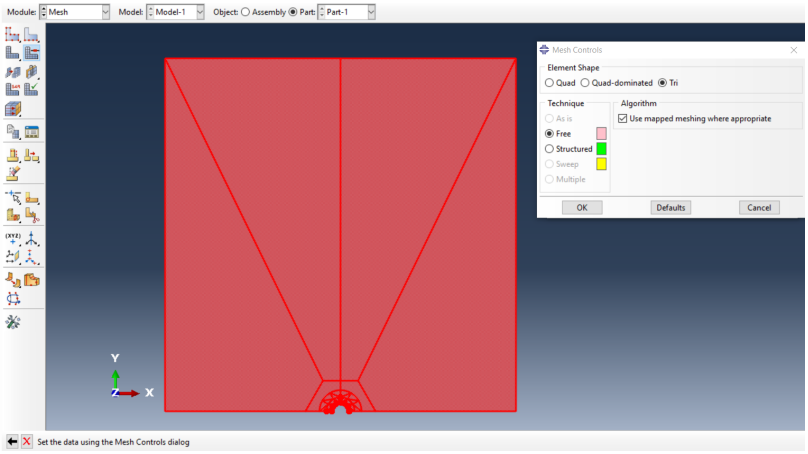


Figure 3.24 Step 8.

```
p = mdb.models['Model-1'].parts['Part-1']
f = p.faces
```

```
    pickedRegions = f.getSequenceFromMask(mask=('[#3fffffff ]', ), )
    p.setMeshControls( regions=pickedRegions,  elemShape=TRI)
```

Script 3.13 Step 10 in Python.

- **Step 11: Part meshing.**. Mesh Module → Mesh Part → Done. Between the half-hexagon and the crowns, we obtain either a symmetric mesh or a mesh in which both halves differ in one vertex connection, that is to say, the orientation of two elementsédges. Inside the crowns, each sub-part is a different element, hence, we always obtain the desired mesh.

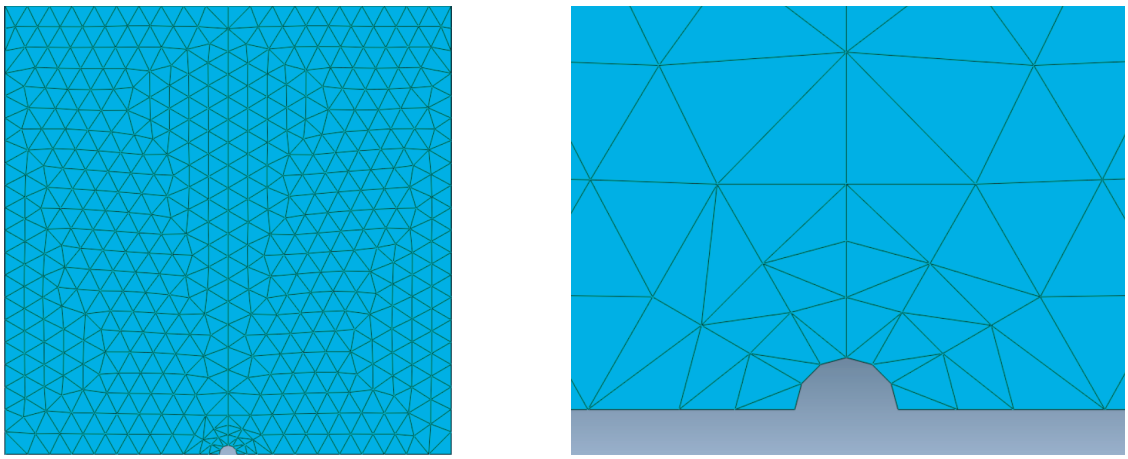


Figure 3.25 Step 8.

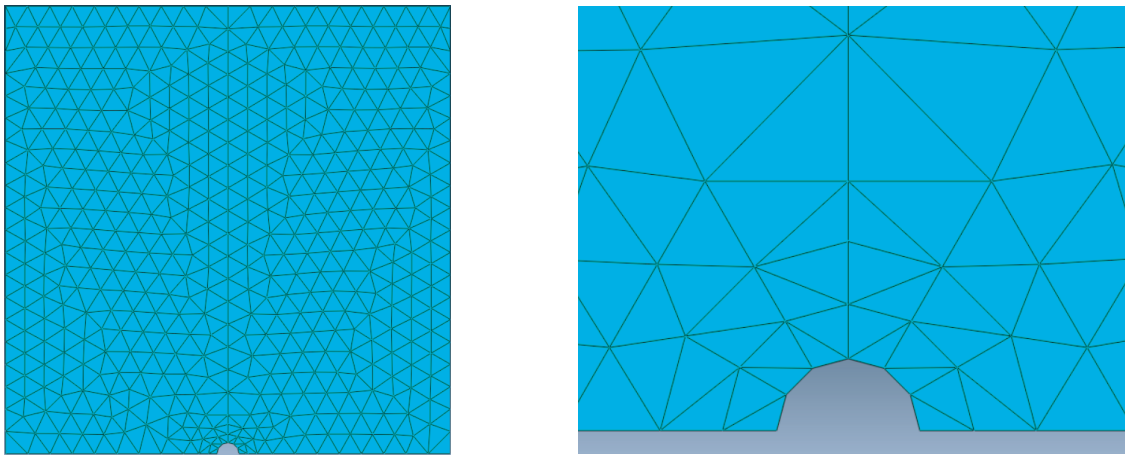


Figure 3.26 Step 8.

```
p = mdb.models['Model-1'].parts['Part-1']
p.generateMesh()
```

Script 3.14 Step 11 in Python.

- **Step 12: Naming, saving and exporting.**. We turn the values of the parameters into strings to compose the naming code. Then, the changing values are introduced in their corresponding spot in the code. We pick only the first 4 or 5 digits of each α value for the code, with the first digit always being a 0 (as a mantissa).

```
a = mdb.models['Model-1'].rootAssembly
a.DatumCsysByDefault(CARTESIAN)
p = mdb.models['Model-1'].parts['Part-1']
a.Instance(name='Part-1-1', part=p, dependent=ON)
alpha1_s=str(alpha1).replace('.', '')
alpha2_s=str(alpha2).replace('.', '')
alpha3_s=str(alpha3).replace('.', '')
tipo1_s=str(tipo1))
tipo2_s=str(tipo2))
```

Script 3.15 Step 12 in Python.

```
!\ ctext [RGB]{252,190,17}{nofelements\_s=str(nofelements))}!
nombre=nofelements\_s+"E\_tipo\_"+tipo1\_s+"\_tipo\_"+tipo2\_s+"\_"+alpha1\_s[0:5]+"\_"+alpha2\_s[0:4]+"\_"+alpha3\
    _s[0:4]
mdb.Job(name=nombre, model='Model-1', description='', type=ANALYSIS,
    atTime=None, waitMinutes=0, waitHours=0, queue=None, memory=90,
    memoryUnits=PERCENTAGE, getMemoryFromAnalysis=True,
     explicitPrecision =DOUBLE_PLUS_PACK, nodalOutputPrecision=FULL,
    echoPrint=OFF, modelPrint=OFF, contactPrint=OFF, historyPrint=OFF,
    userSubroutine='', scratch='', resultsFormat=ODB)
mdb.jobs[nombre].writeInput(consistencyChecking=OFF)
mdb.saveAs(
    pathName='C:/Path/'+nofelements_s+'E/'+nombre)
```

Script 3.16 Step 13 in Python.

3.4 Type 5 crowns

A minor change must be introduced for type 5 crowns, which cannot be built from a circumference due to the vertical line. In this case, the crown should be cut by using the following segment arrangement:

```
s.ArcByCenterEnds(center=(x_ct,y_ct), point1=(x_ct-rco2, y_ct+0), point2=(x_ct+x3[nofelements+1], y_ct+y3[
    nofelements+1]), direction=CLOCKWISE)
s.ArcByCenterEnds(center=(x_ct,y_ct), point1=(x_ct+rco2, y_ct),
    point2=(x_ct+x3[nofelements-1],y_ct+y3[nofelements-1]), direction=COUNTERCLOCKWISE)
s.Line(point1=(x_ct+x3[nofelements+1], y_ct+y3[nofelements+1]), point2=(x_ct+x3[nofelements-1], y_ct+y3[nofelements
    -1]))
```

Script 3.17 Type 5 crown definition in Python.

In order to achieve the desired result:

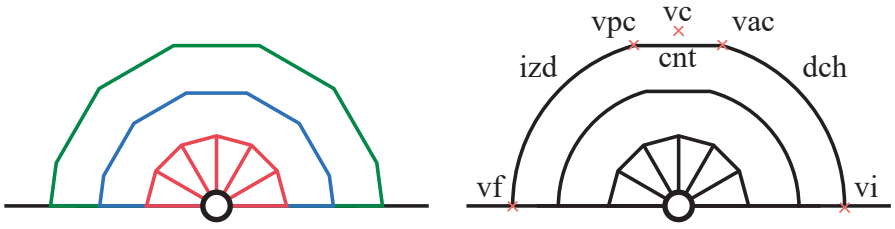


Figure 3.27 Type 5 crown definition.

3.5 Process Improvement

Making a step-by-step process like this required identifying errors separately at each line. That is why this method was chosen. However, after having more experience debugging the code, the process can be time-optimized by doing as follows: instead ofinterchanging different steps that draw segments and erase

regions, using only one step to make all cuts and then remove the SER. The result would make all the macros more interchangeable and save some time. The meshing steps and the result would not change.

3.6 Parametric sweep

This study analyzes:

- The effect of the number of elements around the crack tip.
- Effect of a crown.
- Effect of a second crown.
- The effect of the size proportion between the singular elements and the crowns elements.
- The effect of the mesh type of crowns.

The following diagram describes the parameter sweep.

```
Hexagon_sidelength = 0.1; % 10% of the shell's sidelength
for Number_of_Elements=[4, 6, 8, 12, 24]
 for crown_type_1 = [1, 2, 3, 4, 5]:
  for crown_type_2 = [0, 1, 2, 3, 4, 5];
   for alpha1 =0.05:0.05: max_allowed alpha1_size
    for alpha2=min_allowed_alpha2_size_0.05:max_allowed alpha2_size
     for alpha3=min_allowed_alpha3_size_0.05:max_allowed alpha3_size
        Create part
        Create half hexagon % In this study, it is always the same length, it depends on the parameter
      hexagon_sidelength.
            Coordinates (connected by lines): % A unitary hexagon's vertices coordinates times side-length
                x_hexagon = x_crack_tip +[1,0.5,0,-0.5,-1]* hexagon_sidelength
                y_hexagon = y_crack_tip +[0, sqrt(3)/2, sqrt(3)/2,sqrt(3)/2,0]* hexagon_sidelength
        Create auxiliary lines by segment coordinates:
            Coordinates (connected by lines)
                x_aux = x_crack_tip +[0.5,1*hexagon_sidelength]; x_crack_tip +[-0.5,-1*hexagon_sidelength];
      x_crack_tip +[0, 0]
                y_aux = y_crack_tip +[1, sqrt(3)/2*hexagon_sidelength]; y_crack_tip +[1, sqrt(3)/2*hexagon_sidelength];
       y_crack_tip +[1,0]
        Create circumferences
        Coordinates (by center and radius): % Only the first two ones if there is only one crown
            inner:
                x_center = x_crack_tip, y_center = y_crack_tip; radius = alpha1*hexagon_sidelength
            medium:
                x_center = x_crack_tip, y_center = y_crack_tip; radius = alpha2*hexagon_sidelength
            outer:
                x_center = x_crack_tip, y_center = y_crack_tip; radius = alpha3*hexagon_sidelength
        Hollow inner circumference
        Create crown lines
            Coordinates: discussed in annex % that depend on alpha_1, alpha_2, alpha_3 and type_1 and type_2
        Mesh seeds
        Mesh control
        Mesh part
        Save ".inp"
     end
    end
   end
  end
 end
end
```

Script 3.18 MATLAB pseudo-code.

The process could theoretically employ always the same commands with a few lines changed ifit was not for the internal numeration of the Abaqus elements (edges, vertices, regions, etc.), which differs for every different cut. For instance, the identification code assigned to the singular elements sub-part does not coincide in each case study; so when the region is erased, we will occasionally delete other sub-parts (the crown or exterior regions).
However, for the first crown types 1, 2 and 3, and second crown types 1, 2, and 3; we can (most times) interchange sections of the code.

3.7 Maximum and Minimum Values of Coefficients

The parameters growth by this method results in some invalid combinations. More precisely, when cutting the crown segments, we will find cases like the following:

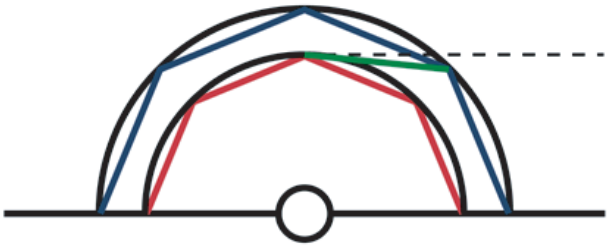


Figure 3.28 Invalid case: segment intersects inner crown.

It would result in an undesired region due to the fact that the cutting line is not tangent to the circumference.

Figure 3.29 Undesired region..

Thus, the minimum value of an $alpha_i$ coefficient depends on the previous circumference's alpha coefficient and on the type of crown. In general:

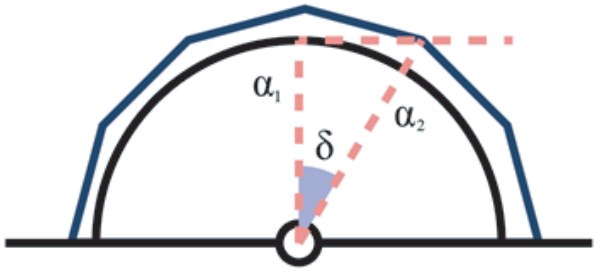


Figure 3.30 Minimum $alpha_2$ that does not cause an undesired region..

$$\delta = \frac{\pi}{\text{number of elements on the inner circumference}} \tag{3.4}$$

Symmetrically, no other undesired region will appear.
As for the upper limit, the loops will automatically dismiss the $alpha_i$ values that causes an alpha value in the last crown higher than maximum allowed. The maximum radius inside the half-hexagon will be the height of the trapezoid:

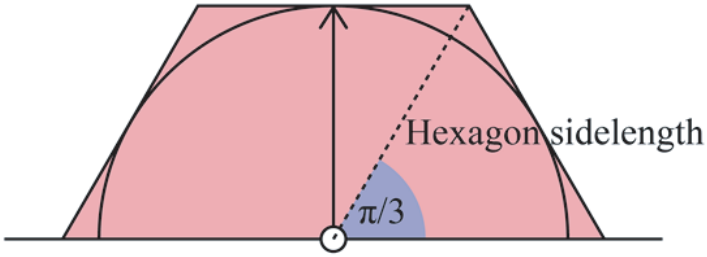


Figure 3.31 Minimum alpha_2 that does not cause an undesired region..

Hence the upper limit will be:

$$\alpha = \mathrm{Hexagon\_sidelength}\cos(\pi/3)/\,\mathrm{Hexagon\_sidelength} = \cos(\pi/3) \approx 0.85 \tag{3.5}$$

These are the upper and lower values of each crown type:

```
Macro_NE_tipo_1_tipo_0():
Macro_NE_tipo_2_tipo_0():
Macro_NE_tipo_3_tipo_0():
    for alpha1 in wp.arange (0.05,0.85,0.05) :
        alpha2min=alpha1*1/(cos(pi/nofelements))
        for alpha2 in wp.arange(alpha2min ,0.85,0.05) :

Macro_6E_tipo_4_tipo_0():
    for alpha1 in wp.arange (0.05,0.85,0.05) :
        alpha2min=alpha1*1/(cos(pi /2/ nofelements))
        for alpha2 in wp.arange(alpha2min ,0.85,0.05) :

Macro_6E_tipo_5_tipo_0():
    for alpha1 in wp.arange (0.05,0.85,0.05) :
        alpha2min=alpha1*1/(cos(pi /2/ nofelements))
      +0.0001
        for alpha2 in wp.arange(alpha2min ,0.85,0.05) :
```

Script 3.19 Bounds for one-crown cases.

A small term is added, which does not affect the sweep, in order to avoid the creation of undesired regions. For 4-element with type 1, 2 and 3 crowns, since the boundaries are more restrictive (and that affected the result), the macro has been changed to include the final cuts for the crown instead of using a circumference, thus allowing to use the lower bound of alpha2min=alpha1*1/(cos(pi/2/nofelements)). The remaining do not need this change.

```

Macro_6E_tipo_1_tipo_1():
Macro_6E_tipo_1_tipo_2():
Macro_6E_tipo_1_tipo_3():
Macro_6E_tipo_2_tipo_1():
Macro_6E_tipo_2_tipo_2():
Macro_6E_tipo_2_tipo_3():
Macro_6E_tipo_3_tipo_1():
Macro_6E_tipo_3_tipo_2():
Macro_6E_tipo_3_tipo_3():
    for alpha1 in wp.arange (0.05,0.85,0.05) :
        alpha2min=alpha1*1/(cos(pi/nofelements))
        for alpha2 in wp.arange(alpha2min ,0.85,0.05) :
            alpha3min=alpha2*1/(cos(pi/nofelements))
            for alpha3 in wp.arange(alpha3min
    ,0.85,0.05) :

Macro_6E_tipo_1_tipo_4():
Macro_6E_tipo_2_tipo_4():
Macro_6E_tipo_3_tipo_4():
Macro_6E_tipo_1_tipo_5():
Macro_6E_tipo_2_tipo_5():
Macro_6E_tipo_3_tipo_5():
    for alpha1 in wp.arange (0.05,0.85,0.05) :
        alpha2min=alpha1*1/(cos(pi/nofelements))
        for alpha2 in wp.arange(alpha2min ,0.85,0.05) :
            alpha3min=alpha2*1/(cos(pi /2/ nofelements))
            for alpha3 in wp.arange(alpha3min
    ,0.85,0.05) :

def Macro_6E_tipo_4_tipo_1():
def Macro_6E_tipo_4_tipo_2():
def Macro_6E_tipo_4_tipo_3():
    for alpha1 in wp.arange (0.05,0.85,0.05) :
        alpha2min=alpha1*1/(cos(pi /2/ nofelements))
        for alpha2 in wp.arange(alpha2min ,0.85,0.05) :
            alpha3min=alpha2*1/(cos(pi/nofelements))
            for alpha3 in wp.arange(alpha3min
    ,0.85,0.05) :

def Macro_6E_tipo_4_tipo_4():
def Macro_6E_tipo_4_tipo_5():
    for alpha1 in wp.arange (0.05,0.85,0.05) :
        alpha2min=alpha1*1/(cos(pi /2/ nofelements))
        for alpha2 in wp.arange(alpha2min ,0.85,0.05) :
            alpha3min=alpha2*1/(cos(pi /2/ nofelements))
            for alpha3 in wp.arange(alpha3min
    ,0.85,0.05) :

def Macro_6E_tipo_5_tipo_1():
def Macro_6E_tipo_5_tipo_2():
def Macro_6E_tipo_5_tipo_3():
    for alpha1 in wp.arange (0.05,0.85,0.05) :
        alpha2min=alpha1*1/(cos(pi /2/ nofelements))
      +0.0001
        for alpha2 in wp.arange(alpha2min ,0.85,0.05) :
            alpha3min=alpha2*1/(cos(pi/nofelements))
            for alpha3 in wp.arange(alpha3min
    ,0.85,0.05) :

def Macro_6E_tipo_5_tipo_4():
def Macro_6E_tipo_5_tipo_5():
```

```
57      for alpha1 in wp.arange (0.05,0.85,0.05) :
58          alpha2min=alpha1*1/(cos(pi/2/nofelements))
          +0.0001
59          for alpha2 in wp.arange(alpha2min ,0.85,0.05) :
60              alpha3min=alpha2*1/(cos(pi/2/nofelements))
61              for alpha3 in wp.arange(alpha3min
          ,0.85,0.05) :
```

Script 3.20 Bounds for two-crown cases.

For 12-and-24-elements-type-5 crown, Abaqusínternal numeration varies when $alpha_2$ or $alpha_3$ are much lower than 0.1, thus we should start by $\alpha_1 = 0.2$. The alternative to include those values is making the sweep in Abaqus Students (where such problem is absent) or making a macro for each different case. However, since in the other case studies those $alpha_1$ values do not provide valid conclusions, we will not add those values.

3.8 Macro Example

In order to save time, two-crown macros are programmed in a slightly different order than previously defined. Since we have a one-crown macro previously programmed, we can use it as the first part of the two-crown macro by adding the second crown after the cutting of the first crown segments (and adding the new seed meshes). The only changes would be adding the third circumference and second crown cuts after the first crown cut.
The whole real code for two crowns is here presented, which hence includes the code for one crown almost completely. Note that the Object t indicates the opposite values of the origin coordinates of the system in which we are sketching.

```
1       tipo1=1
2       tipo2=1
3   #____________________# Crown definition, vertices coordinates
4       nofelements=6   # Change
5       sidelength=0.1  # Change
6
7       # Hexagon coordinates:
8       xpaux=sidelength*wp.array ([1,0.5,0,-0.5,-1])
9       ypaux=sidelength*wp.array([0,0.866025403784439,0.866025403784439,0.866025403784439,0])
10
11      # Loop for type_1_type_1 crowns:
12      for alpha1 in wp.arange (0.05,0.85,0.05) :
13          alpha2min=alpha1*1/(cos(pi/nofelements))
14          for alpha2 in wp.arange(alpha2min ,0.85,0.05) :
15              alpha3min=alpha2*1/(cos(pi/nofelements))
16              for alpha3 in wp.arange(alpha3min ,0.85,0.05) :
17                  N   = 2*nofelements
18                  n   = list (range(0, nofelements+1))
19                  n2  = list (range(0, nofelements+1))
20                  n3  = list (range(0, nofelements+1))
21                  rci = sidelength*alpha1
22                  rco = sidelength*alpha2
23                  rco2=sidelength*alpha3
24                  x1  = list (range(0, nofelements+1))
25                  y1  = list (range(0, nofelements+1))
26                  x2  = list (range(0, nofelements+1))
27                  y2  = list (range(0, nofelements+1))
28                  x3  = list (range(0, nofelements+1))
29                  y3  = list (range(0, nofelements+1))
30                  for i in range(0,nofelements+1):
31                      x1[i]=rci*cos(2*pi*n[i]/N)
32                      y1[i]=rci*sin(2*pi*n[i]/N)
33                  for i in range(0,nofelements+1):
34                      x2[i]=rco*cos(2*pi*n2[i]/(N))
35                      y2[i]=rco*sin(2*pi*n2[i]/(N))
36                  for i in range(0,nofelements+1):
37                      x3[i]=rco2*cos(2*pi*n3[i]/(N))
38                      y3[i]=rco2*sin(2*pi*n3[i]/(N))
39
40  #____________________# Step01
41                  Mdb()
42                  s = mdb.models['Model-1'].ConstrainedSketch(name='\_\ _profile \_\_', sheetSize=2.0)
43                  g, v, d, c = s.geometry, s.vertices , s.dimensions, s.constraints
```

```
                    s.setPrimaryObject(option=STANDALONE)
                    s.rectangle(point1=(-0.5, 0.0), point2=(0.5, 1.0))
                    p = mdb.models['Model-1'].Part(name='Part-1', dimensionality=TWO\_D\_PLANAR,
                        type=DEFORMABLE\_BODY)
                    p = mdb.models['Model-1'].parts['Part-1']
                    p.BaseShell(sketch=s)
                    s.unsetPrimaryObject()
                    p = mdb.models['Model-1'].parts['Part-1']

#___________________# Step02
                    p = mdb.models['Model-1'].parts['Part-1']
                    f1, e1, d2 = p.faces, p.edges, p.datums
                    t = p.MakeSketchTransform(sketchPlane=f1[0], sketchPlaneSide=SIDE1, origin=(
                        0.0, 0.5, 0.0))
                    s = mdb.models['Model-1'].ConstrainedSketch(name='\_\ _profile \_\_', sheetSize=2.82,
                        gridSpacing=0.07, transform=t)
                    g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
                    s.setPrimaryObject(option=SUPERIMPOSE)
                    p = mdb.models['Model-1'].parts['Part-1']
                    p.projectReferencesOntoSketch(sketch=s, filter=COPLANAR\_EDGES)
                    s.Line(point1=(0.0, -0.51), point2=(0.0+xpaux[0], -0.5+ypaux[0]))
                    s.Line(point1=(0.0+xpaux[0], -0.5+ypaux[0]), point2=(0.0+xpaux[1], -0.5+ypaux[1]))
                    s.Line(point1=(0.0+xpaux[1], -0.5+ypaux[1]), point2=(0.0+xpaux[2], -0.5+ypaux[2]))
                    s.Line(point1=(0.0+xpaux[2], -0.5+ypaux[2]), point2=(0.0+xpaux[3], -0.5+ypaux[3]))
                    s.Line(point1=(0.0+xpaux[3], -0.5+ypaux[3]), point2=(0.0+xpaux[4], -0.5+ypaux[4]))
                    s.Line(point1=(0.0+xpaux[4], -0.5+ypaux[4]), point2=(0.0, -0.51))
                    s.Line(point1=(0.5, 0.5), point2=(0.0+xpaux[1], -0.5+ypaux[1]))
                    s.Line(point1=(-0.5, 0.5), point2=(0.0+xpaux[3], -0.5+ypaux[3]))
                    s.Line(point1=(0.0, -0.5), point2=(0.0, 0.5))
                    p = mdb.models['Model-1'].parts['Part-1']
                    f = p.faces
                    pickedFaces = f.getSequenceFromMask(mask=('[#1 ]', ), )
                    e, d1 = p.edges, p.datums
                    p.PartitionFaceBySketch(faces=pickedFaces, sketch=s)
                    s.unsetPrimaryObject()

#___________________# Step03
                    p = mdb.models['Model-1'].parts['Part-1']
                    f, e1, d2 = p.faces, p.edges, p.datums
                    t = p.MakeSketchTransform(sketchPlane=f[1], sketchPlaneSide=SIDE1, origin=(
                        0.097222, 0.096225, 0.0))
                    s1 = mdb.models['Model-1'].ConstrainedSketch(name='\_\ _profile \_\_',
                        sheetSize=1.11, gridSpacing=0.02, transform=t)
                    g, v, d, c = s1.geometry, s1.vertices, s1.dimensions, s1.constraints
                    s1.setPrimaryObject(option=SUPERIMPOSE)
                    p = mdb.models['Model-1'].parts['Part-1']
                    p.projectReferencesOntoSketch(sketch=s1, filter=COPLANAR\_EDGES)
                    s1.sketchOptions.setValues(gridOrigin=(-0.097222, -0.096225))
                    s1.CircleByCenterPerimeter(center=(-0.097222, -0.096225), point1=(-0.097222+rci, -0.096225))
                    s1.CircleByCenterPerimeter(center=(-0.097222, -0.096225), point1=(-0.097222+rco, -0.096225))
                    p = mdb.models['Model-1'].parts['Part-1']
                    f = p.faces
                    pickedFaces = f.getSequenceFromMask(mask=('[#a ]', ), )
                    e, d1 = p.edges, p.datums
                    p.PartitionFaceBySketch(faces=pickedFaces, sketch=s1)
                    s1.unsetPrimaryObject()

#___________________# Step04
                    p = mdb.models['Model-1'].parts['Part-1']
                    f1 = p.faces
                    p.RemoveFaces(faceList = f1[5:6]+f1[7:8], deleteCells=False)

#___________________# Step05
                    p = mdb.models['Model-1'].parts['Part-1']
                    f, e1, d2 = p.faces, p.edges, p.datums
                    t = p.MakeSketchTransform(sketchPlane=f[1], sketchPlaneSide=SIDE1, origin=(
                        -0.082041, 0.082041, 0.0))
                    s = mdb.models['Model-1'].ConstrainedSketch(name='\_\ _profile \_\_', sheetSize=0.91,
                        gridSpacing=0.02, transform=t)
                    g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
                    s.setPrimaryObject(option=SUPERIMPOSE)
```

```
                p = mdb.models['Model-1'].parts['Part-1']
                p.projectReferencesOntoSketch(sketch=s, filter=COPLANAR\_EDGES)
                s.Line(point1=(0.082041+x2[0],-0.082041+y2[0]), point2=(0.082041+x1[1],-0.082041+y1[1]))
                s.Line(point1=(0.082041+x1[1],-0.082041+y1[1]), point2=(0.082041+x2[1],-0.082041+y2[1]))
                s.Line(point1=(0.082041+x2[1],-0.082041+y2[1]), point2=(0.082041+x1[2],-0.082041+y1[2]))
                s.Line(point1=(0.082041+x1[2],-0.082041+y1[2]), point2=(0.082041+x2[2],-0.082041+y2[2]))
                s.Line(point1=(0.082041+x2[2],-0.082041+y2[2]), point2=(0.082041+x1[3],-0.082041+y1[3]))
                s.Line(point1=(0.082041+x1[3],-0.082041+y1[3]), point2=(0.082041+x2[3],-0.082041+y2[3]))
                s.Line(point1=(0.082041+x1[3],-0.082041+y1[3]), point2=(0.082041+x2[4],-0.082041+y2[4]))
                s.Line(point1=(0.082041+x2[4],-0.082041+y2[4]), point2=(0.082041+x1[4],-0.082041+y1[4]))
                s.Line(point1=(0.082041+x1[4],-0.082041+y1[4]), point2=(0.082041+x2[5],-0.082041+y2[5]))
                s.Line(point1=(0.082041+x2[5],-0.082041+y2[5]), point2=(0.082041+x1[5],-0.082041+y1[5]))
                s.Line(point1=(0.082041+x1[5],-0.082041+y1[5]), point2=(0.082041+x2[6],-0.082041+y2[6]))
                p = mdb.models['Model-1'].parts['Part-1']
                f = p.faces
                pickedFaces = f.getSequenceFromMask(mask=('[#a ]', ), )
                e, d1 = p.edges, p.datums
                p.PartitionFaceBySketch(faces=pickedFaces, sketch=s)
                s.unsetPrimaryObject()

#_____________________# Step06
                f, e, d = p.faces, p.edges, p.datums
                t = p.MakeSketchTransform(sketchPlane=f[10], sketchPlaneSide=SIDE1, origin=(
                    -0.049686, 0.04895, 0.0))
                s = mdb.models['Model-1'].ConstrainedSketch(name='\_\ _profile \_\_',
                    sheetSize=0.264, gridSpacing=0.006, transform=t)
                g, v, d1, c = s.geometry, s.vertices, s.dimensions, s.constraints
                s.sketchOptions.setValues(decimalPlaces=3)
                s.setPrimaryObject(option=SUPERIMPOSE)
                p = mdb.models['Model-1'].parts['Part-1']
                p.projectReferencesOntoSketch(sketch=s, filter=COPLANAR\_EDGES)
                s.sketchOptions.setValues(gridOrigin=(0.049686, -0.04895))
                s.CircleByCenterPerimeter(center=(0.049686, -0.04895), point1=(0.049686+rco2, -0.04895))
                s.CoincidentConstraint(entity1=v[25], entity2=g[36], addUndoState=False)
                p = mdb.models['Model-1'].parts['Part-1']
                f = p.faces
                pickedFaces = f.getSequenceFromMask(mask=('[#1400 ]', ), )
                e1, d2 = p.edges, p.datums
                p.PartitionFaceBySketch(faces=pickedFaces, sketch=s)
                s.unsetPrimaryObject()

#_____________________# Step07
                f1, e, d = p.faces, p.edges, p.datums
                t = p.MakeSketchTransform(sketchPlane=f1[0], sketchPlaneSide=SIDE1, origin=(
                    0.047228, 0.047228, 0.0))
                s1 = mdb.models['Model-1'].ConstrainedSketch(name='\_\ _profile \_\_',
                    sheetSize=0.41, gridSpacing=0.01, transform=t)
                g, v, d1, c = s1.geometry, s1.vertices, s1.dimensions, s1.constraints
                s1.sketchOptions.setValues(decimalPlaces=3)
                s1.setPrimaryObject(option=SUPERIMPOSE)
                p = mdb.models['Model-1'].parts['Part-1']
                p.projectReferencesOntoSketch(sketch=s1, filter=COPLANAR\_EDGES)
                s1.sketchOptions.setValues(gridOrigin=(-0.047228, -0.047228))
                s1.Line(point1=(-0.047228+x3[0],-0.047228+y3[0]), point2=(-0.047228+x2[1],-0.047228+y2[1]))
                s1.Line(point1=(-0.047228+x2[1],-0.047228+y2[1]), point2=(-0.047228+x3[1],-0.047228+y3[1]))
                s1.Line(point1=(-0.047228+x3[1],-0.047228+y3[1]), point2=(-0.047228+x2[2],-0.047228+y2[2]))
                s1.Line(point1=(-0.047228+x2[2],-0.047228+y2[2]), point2=(-0.047228+x3[2],-0.047228+y3[2]))
                s1.Line(point1=(-0.047228+x3[2],-0.047228+y3[2]), point2=(-0.047228+x2[3],-0.047228+y2[3]))
                s1.Line(point1=(-0.047228+x2[3],-0.047228+y2[3]), point2=(-0.047228+x3[3],-0.047228+y3[3]))
                s1.Line(point1=(-0.047228+x2[3],-0.047228+y2[3]), point2=(-0.047228+x3[4],-0.047228+y3[4]))
                s1.Line(point1=(-0.047228+x3[4],-0.047228+y3[4]), point2=(-0.047228+x2[4],-0.047228+y2[4]))
                s1.Line(point1=(-0.047228+x2[4],-0.047228+y2[4]), point2=(-0.047228+x3[5],-0.047228+y3[5]))
                s1.Line(point1=(-0.047228+x3[5],-0.047228+y3[5]), point2=(-0.047228+x2[5],-0.047228+y2[5]))
                s1.Line(point1=(-0.047228+x2[5],-0.047228+y2[5]), point2=(-0.047228+x3[6],-0.047228+y3[6]))
                p = mdb.models['Model-1'].parts['Part-1']
                f = p.faces
                pickedFaces = f.getSequenceFromMask(mask=('[#1001 ]', ), )
                e1, d2 = p.edges, p.datums
                p.PartitionFaceBySketch(faces=pickedFaces, sketch=s1)
                s1.unsetPrimaryObject()
                del mdb.models['Model-1'].sketches['\_\ _profile \_\_']
```

```
186
187 #____________________# Step08 for case a.
188                     p = mdb.models['Model-1'].parts['Part-1']
189                     e = p.edges
190                     pickedEdges = e.getSequenceFromMask(mask=('[#e1fffff  # fff  ]', ), )
191                     p.seedEdgeByNumber(edges=pickedEdges, number=1, constraint=FIXED)
192                     p = mdb.models['Model-1'].parts['Part-1']
193                     f = p.faces
194                     pickedRegions = f.getSequenceFromMask(mask=('[#3 fffffff  ]', ), )
195                     p.setMeshControls(regions=pickedRegions, elemShape=TRI)
196                     p = mdb.models['Model-1'].parts['Part-1']
197                     p.seedPart(size=0.05, deviationFactor=0.1, minSizeFactor=0.1)
198                     p = mdb.models['Model-1'].parts['Part-1']
199                     f = p.faces
200                     pickedRegions = f.getSequenceFromMask(mask=('[#3 fffffff  ]', ), )
201                     p.generateMesh(regions=pickedRegions)
202
203 #____________________# Step09, Step 10 for case b.
204                     p = mdb.models['Model-1'].parts['Part-1']
205                     p = mdb.models['Model-1'].parts['Part-1']
206                     f = p.faces
207                     pickedRegions = f.getSequenceFromMask(mask=('[#25000800 ]', ), )
208                     p.deleteMesh(regions=pickedRegions)
209                     p = mdb.models['Model-1'].parts['Part-1']
210                     e = p.edges
211                     pickedEdges = e.getSequenceFromMask(mask=('[#8000000 #20000 ]', ), )
212                     p.seedEdgeByNumber(edges=pickedEdges, number=3)
213                     p = mdb.models['Model-1'].parts['Part-1']
214                     f = p.faces
215                     pickedRegions = f.getSequenceFromMask(mask=('[#18000000 ]', ), )
216                     p.deleteMesh(regions=pickedRegions)
217                     p = mdb.models['Model-1'].parts['Part-1']
218                     e = p.edges
219                     pickedEdges = e.getSequenceFromMask(mask=('[#16000000 #50000 ]', ), )
220                     p.seedEdgeByNumber(edges=pickedEdges, number=1)
221
222 #____________________# Step11
223                     p = mdb.models['Model-1'].parts['Part-1']
224                     p.generateMesh()
225
226 #____________________# Step12
227                     p = mdb.models['Model-1'].parts['Part-1']
228                     p.generateMesh()
229                     a = mdb.models['Model-1'].rootAssembly
230                     a.DatumCsysByDefault(CARTESIAN)
231                     p = mdb.models['Model-1'].parts['Part-1']
232                     a.Instance(name='Part-1-1', part=p, dependent=ON)
233                     alpha1\_s=str(alpha1).replace('.', '')
234                     alpha2\_s=str(alpha2).replace('.', '')
235                     alpha3\_s=str(alpha3).replace('.', '')
236                     tipo1\_s=str(tipo1)
237                     tipo2\_s=str(tipo2)
238                     nofelements\_s=str(nofelements)
239                     nombre=nofelements\_s+"E\_tipo\_"+tipo1\_s+"\_tipo\_"+tipo2\_s+"\_"+alpha1\_s[0:5]+"\_"+alpha2\_s
        [0:4]+"\_"+alpha3\_s[0:4]
240                     mdb.Job(name=nombre, model='Model-1', description='', type=ANALYSIS,
241                         atTime=None, waitMinutes=0, waitHours=0, queue=None, memory=90,
242                         memoryUnits=PERCENTAGE, getMemoryFromAnalysis=True,
243                          explicitPrecision=DOUBLE\_PLUS\_PACK, nodalOutputPrecision=FULL,
244                         echoPrint=OFF, modelPrint=OFF, contactPrint=OFF, historyPrint=OFF,
245                         userSubroutine='', scratch='', resultsFormat=ODB)
246                     mdb.jobs[nombre].writeInput(consistencyChecking=OFF)
247                     mdb.saveAs(
248                         pathName=Path\_name+nofelements\_s+'E/'+nombre)
```

Script 3.21 Macro example.

3.9 Simulation

For the case simulation we will be using a slightly modified version of [3], [5], [17]. Each of the program's sections is discussed in its comments. The new program additions are the automatic imposition of boundary conditions, which is discussed in depth. Without it, the analysis of such an amount of meshes would not be feasible.

```
%% Main loop. It must be in the same folder as the folders containing the .inp files.

clear all, close all, clc

%% This program solves our case study and gives displacements in the nodes by FEM method as outcome.

%% Identification of the folder with .inp files.
id='4E_tipo_1_tipo_0'; %% Name of the folder

%% Input (write values):
hex_sidelength =0.1;            % Half-hexagon side-length.
typeelement=1;                  % 1: with singular elements // 0: without singular elements
vct=[0.00 0.00 0.00];           % Vector with crack tip coordinates.
G=1; k_robin=1; NUM=400;

%% Initialize variables.
[name_index, file_index ,alpha1_index,alpha2_index,alpha3_index, desplazamientos_calculo ,coordenadas_calculo ,
        singular_FE_sidelength_index ,new_entrance_inp_index] = step_01_set_variables ();

%% Obtener parámetros del nombre
[nofelements, nofcrowns, tipo1, tipo2, separadores] = step_02_obtain_parameters(id);

%% Name index
[alpha1_index,alpha2_index,alpha3_index, singular_FE_sidelength_index , file_index ,name_index,ncasos] =
        step_03_name_archives(nofelements,nofcrowns,tipo1,tipo2,hex_sidelength,id,name_index, file_index ,
        singular_FE_sidelength_index ,alpha1_index,alpha2_index,alpha3_index);

%% Loop to analyze every case study
for s=1:size(name_index,1)
close all;

%% Change in the work directory (change it to fit the PC in use)
cd([ 'C:\Users\Usuario\Desktop\TFG (Avances y Bibliografia )\Macros\',id(1:separadores(1)-1),'\',id]);

%% Specifying the values of certain parameters to fit the case index
alpha1=alpha1_index(:,s);
SE_sidelength= singular_FE_sidelength_index(s);
 file = file_index (s ,:) ;
inp_malla= fileread ( file );

%% Obtain meshes
cd([ 'C:\Users\Usuario\Desktop\TFG (Avances y Bibliografia )\Macros\Codigo Matlab']);
[coordenadas,coordenadasN5,coordenadas5, triangulos , triangulos5 ]=step_04_mesher(inp_malla, nofelements,SE_sidelength,
        alpha1,typeelement, vct);

%% Specifying boundary conditions
[neumann,neumann5,dirichlet, dirichlet5 ,robin,robin5,nodos_calculo]=step_05_boundary_conditions(coordenadas, vct ,
        triangulos5 );

%% Counting elements
[numNodos,numTriangulos,numTriangulos5,numLadosNeu,numLadosNeu5,numLadosDir,numLadosDir5,numLadosRob,
        numLadosRob5]=step_06_count_elements(coordenadas,triangulos,triangulos5,neumann,neumann5,dirichlet,dirichlet5,
        robin,robin5);

%% K_mn matrix of the singular elements
RobMat=step_07_robin_matrix();

%% Variable initialization
[K,b,KROBIN,u] = step_08_initialize_system_matrixes (numNodos);

%% Stiffness matrix of the system
K = step_09_stiffness_matrix (coordenadas, triangulos , triangulos5 ,numTriangulos,numTriangulos5,G,K);
```

```
56
57 %% Generation of matrices to assamble alongside stiffness matrix due to Robin conditions
58 KROBIN = step_10_robin_stiffness_matrix(numLadosRob,numLadosRob5,robin,robin5,coordenadas,k_robin,RobMat,
       numTriangulos,triangulos,KROBIN,NUM);
59
60 %% Assemble Robin conditions in the stiffness matrix:
61 K=K+KROBIN;
62
63 %% Neumann conditions
64 b = step_11_neumann_conditions(b,numLadosNeu,numLadosNeu5,neumann,neumann5,coordenadas,numTriangulos,RobMat,
       numTriangulos5,triangulos,triangulos5);
65
66 %% Dirichlet conditions
67 [b,Dir,Dir5] = step_12_dirichlet_conditions (b,numLadosDir,numLadosDir5,dirichlet, dirichlet5 ,coordenadas,
       numTriangulos,RobMat,numTriangulos5,triangulos, triangulos5 );
68
69 %% Independent vector
70 b=b-K*u;
71
72 %% Solution
73 nodosDirTotales=[Dir;Dir5]; %forma una matriz con todos los nodos con condiciones Dirichlet
74 nodosLibres= setdiff (1:numNodos,nodosDirTotales); %excluye a los nodos con condiciones Dirichlet
75 u(nodosLibres)=K(nodosLibres,nodosLibres)\b(nodosLibres);
76
77 %% Displacement along the crack
78 desplazamientos_calculo =[ desplazamientos_calculo ; u(nodos_calculo) '];
79 coordenadas_calculo=[coordenadas_calculo; coordenadas(nodos_calculo,1) '];
80 end
81
82 %% Regression
83 [ ajuste , ajuste_2 , slope_index ,max_2nd_crown_change,casos_buenos] = step_13_numerical_regressions(alpha1_index,
       alpha2_index,alpha3_index, desplazamientos_calculo , coordenadas_calculo , tipo1 , tipo2 , id , separadores );
84
85 %% Saving
86 cd([ 'C:\Users\Usuario\Desktop\TFG (Avances y Bibliografia )\Macros\' , id (1: separadores (1)-1), '\' , id ]);
87 save(id);
88 cd([ 'C:\Users\Usuario\Desktop\TFG (Avances y Bibliografia )\Macros\Codigo Matlab']);
89
90 end
```

Script 3.22 Step 00 in MATLAB, main body.

3.9.1 step_01_set_variables

```
1 function [name_index, file_index ,alpha1_index,alpha2_index,alpha3_index, desplazamientos_calculo , coordenadas_calculo ,
       singular_FE_sidelength_index ,new_entrance_inp_index] = step_01_set_variables ()
2 %% This code initializes vectors that will be used in the program.
3 name_index=[];
4 file_index =[];
5 alpha1_index =[];
6 alpha2_index =[];
7 alpha3_index =[];
8 desplazamientos_calculo =[];
9 coordenadas_calculo =[];
10 singular_FE_sidelength_index =[];
11 new_entrance_inp_index=[];
12 end
```

Script 3.23 Step 01 in MATLAB.

3.9.2 step_02_obtain_parameters

```
1 function [nofelements, nofcrowns, tipo1 , tipo2 , separadores ] = step_02_obtain_parameters (id)
2 %% This code finds the crown type data by the naming code
3 separadores = strfind (id , '_');
```

```
nofelements=str2num(id(1:separadores(1)-2));
if str2num(id(separadores(end)+1:end))~=0
    nofcrowns=2;
else
    nofcrowns=1;
end
tipo1=str2num(id(separadores(2)+1:separadores(3)-1));
tipo2=str2num(id(separadores(4)+1:end));
```

Script 3.24 Step 02 in MATLAB.

3.9.3 step_03_name_archives

A reduced version of the code is here presented. The complete code in in the appendix. The scripts is explained in the commented lines.

```

function [alpha1_index,alpha2_index,alpha3_index, singular_FE_sidelength_index , file_index ,name_index,ncasos] =
    step_03_name_archives(nofelements,nofcrowns,tipo1 , tipo2 , hex_sidelength , id ,name_index, file_index ,
    singular_FE_sidelength_index ,alpha1_index,alpha2_index,alpha3_index, separadores )
cd(['C:\Users\Usuario\Desktop\TFG (Avances y Bibliografia )\Macros\',id(1:separadores(1)-1),'\'],id]);
new_entrance=[];

%% This code generates the name index that we will use to identify files and read them. It also generates an index
    of alpha_1, alpha_2 and alpha_3 by repeating Abaqus' loops with the same values. Since their names were
    originally created in Abaqus, the strings generated in MATLAB will probably differ in the number of zeros
    added or in one decimal due to rounding. That is why some lines are added to fit the format to Abaqus'. The
    limits are those indicated previously for each crown type. The process is as follows: we turn the alpha
    coefficients into strings with the most likely format. If that file is not found, we know that as much we
    have to add or subtract 0.0001, which is the format limit (four decimals precision). The configurations are
    several: alpha_2 can augment, decrease or remain equal, but if there is a second crown, alpha_3 can
    simultaneously do the same. All the possibilities are explored, and the first one to be found is chosen.

%% 1 Crown:

%% type_1_type_0, type_2_type_0, type_3_type_0
if nofcrowns==1 && (tipo1==1 || tipo1==2 || tipo1==3)
    alpha1_in=0.05;
    alpha1_in_plus=0;
    alpha2_in_n=1/cos(pi/nofelements);
    alpha2_in_plus=0.05;
end

%% type_4_type_0
if nofcrowns==1 && tipo1==4
    alpha1_in=0.05;
    alpha1_in_plus=0;
    alpha2_in_n=1/cos(pi/2/nofelements);
    alpha2_in_plus=0;
end
%% type_5_type_0 for 4, 6 and 8 elements.
if nofcrowns==1 && tipo1==5 && (nofelements==4 || nofelements==6 || nofelements==8)
    alpha1_in=0.05;
    alpha1_in_plus=0;
    alpha2_in_n=1/cos(pi/2/nofelements);
    alpha2_in_plus=0.0001;
end

%% Two crowns

if nofcrowns==2 && (tipo1==1 || tipo1==2 || tipo1==3) && (tipo2==1 || tipo2==2 || tipo2==3)
    alpha1_in=0.05;
    alpha1_in_plus=0;
    alpha2_in_n=1/(cos(pi/nofelements));
    alpha2_in_plus=0.05;
    alpha3_in_n=1/(cos(pi/nofelements));
    alpha3_in_plus=0.05;
end

```

```
if   nofcrowns==2 && (tipo1==1 || tipo1==2 || tipo1==3) && tipo2==4
     alpha1_in=0.05;
     alpha1_in_plus=0;
     alpha2_in_n=1/(cos(pi/nofelements));
     alpha2_in_plus=0;
     alpha3_in_n=1/(cos(pi/2/nofelements));
     alpha3_in_plus=0;
end

if   nofcrowns==2 && (tipo1==1 || tipo1==2 || tipo1==3) && tipo2==5
     alpha1_in=0.05;
     alpha1_in_plus=0;
     alpha2_in_n=1/(cos(pi/nofelements));
     alpha2_in_plus=0.05;
     alpha3_in_n=1/(cos(pi/2/nofelements));
     alpha3_in_plus=0.05;
end

%% One-crown
if   nofcrowns==1,
     for alpha1=alpha1_in+alpha1_in_plus :0.05:0.85
          for alpha2=alpha1*alpha2_in_n+alpha2_in_plus :0.05:0.85
               singular_FE_sidelength=alpha1*hex_sidelength;
               alpha1_s=num2str((alpha1),4);
               alpha1_s=strrep(alpha1_s,'.','');
               alpha2_s=num2str((alpha2),'%2.15f');
               alpha2_s=strrep(alpha2_s,'.','');
               alpha2_s=alpha2_s(1:4);
               Files=dir(['*','_',alpha1_s,'_',alpha2_s(1:end),'*.inp']);
               if isempty(Files)
                    alpha2_s=strrep(num2str(alpha2+0.0001,'%2.15f'),'.','');
                    alpha2_s=alpha2_s(1:4);
                    Files=[];
                    Files=dir(['*','_',alpha1_s,'_',alpha2_s(1:end),'*.inp']);
                    if isempty(Files)
                         alpha2_s=strrep(num2str(alpha2-0.0001,'%2.15f'),'.','');
                         alpha2_s=alpha2_s(1:4);
                    end
               end
               new_entrance=[id alpha1_s '_' alpha2_s];
               new_entrance_inp=[id '_' alpha1_s '_' alpha2_s '.inp'];
               name_index=strvcat(name_index, new_entrance);
               file_index=strvcat(file_index,new_entrance_inp);
               singular_FE_sidelength_index=[singular_FE_sidelength_index; singular_FE_sidelength];
               alpha1_index=[alpha1_index,alpha1];
               alpha2_index=[alpha2_index,alpha2];
          end
     end
end

%% Two-crowns
if   nofcrowns==2
     for alpha1=alpha1_in+alpha1_in_plus :0.05:0.85
          for alpha2=alpha1*alpha2_in_n+alpha2_in_plus :0.05:0.85
               for alpha3=alpha2*alpha3_in_n+alpha3_in_plus :0.05:0.85
                    % Singular element side-length
                    singular_FE_sidelength=alpha1*hex_sidelength;

                    % alpha1
                    alpha1_s=num2str((alpha1),4);
                    alpha1_s=strrep(alpha1_s,'.','');

                    % alpha2
                    alpha2_s=num2str((alpha2),'%2.15f');
                    alpha2_s=strrep(alpha2_s,'.','');
                    alpha2_s=alpha2_s(1:4);

                    % alpha3
                    alpha3_s=num2str((alpha3),'%2.15f');
                    alpha3_s=strrep(alpha3_s,'.','');
                    if length(alpha3_s)>3, alpha3_s=alpha3_s(1:4); else, alpha3_s=alpha3_s(1:end); end
```

```
115
116                 % = alpha2, = alpha3
117                 Files=dir(['*','_',alpha1_s,'_',alpha2_s(1:end),'_',alpha3_s(1:end),'*.inp']);
118                 if isempty(Files),
119                     % = alpha2, + alpha3.
120                     alpha2_aux=str2num([alpha2_s(1),'.',alpha2_s(2:end)]);
121                     alpha2_saux=strrep(num2str(alpha2_aux+0.001,'%2.15f'),'.','');
122                     alpha3_aux=str2num([alpha3_s(1),'.',alpha3_s(2:end)]);
123                     alpha3_saux=strrep(num2str(alpha3_aux+0.001,'%2.15f'),'.','');
124                     alpha3_saux=alpha3_saux(1:4);
125                     Files=dir(['*','_',alpha1_s,'_',alpha2_saux(1:end),'_',alpha3_saux(1:end),'*.inp']);
126                     if ~isempty(Files),
127                         alpha2_s=alpha2_saux;
128                         alpha3_s=alpha3_saux;
129                     else
130                     % = alpha2, -alpha3.
131                     alpha2_aux=str2num([alpha2_s(1),'.',alpha2_s(2:end)]);
132                     alpha3_aux=str2num([alpha3_s(1),'.',alpha3_s(2:end)]);
133                     alpha3_saux=strrep(num2str(alpha3_aux-0.001,'%2.15f'),'.','');
134                     alpha3_saux=alpha3_saux(1:4);
135                     Files=dir(['*','_',alpha1_s,'_',alpha2_saux(1:end),'_',alpha3_saux(1:end),'*.inp']);
136                     if ~isempty(Files),
137                         alpha2_s=alpha2_saux;
138                         alpha3_s=alpha3_saux;
139                     else
140                     % + alpha2, = alpha3
141                     alpha2_aux=str2num([alpha2_s(1),'.',alpha2_s(2:end)]);
142                     alpha3_aux=str2num([alpha3_s(1),'.',alpha3_s(2:end)]);
143                     alpha2_saux=strrep(num2str(alpha2_aux+0.001,'%2.15f'),'.','');
144                     alpha2_saux=alpha2_saux(1:4);
145                     Files=dir(['*','_',alpha1_s,'_',alpha2_saux(1:end),'_',alpha3_saux(1:end),'*.inp']);
146                     if ~isempty(Files),
147                         alpha2_s=alpha2_saux;
148                         alpha3_s=alpha3_saux;
149                     else
150                     % + alpha2, + alpha3
151                     alpha2_aux=str2num([alpha2_s(1),'.',alpha2_s(2:end)]);
152                     alpha3_aux=str2num([alpha3_s(1),'.',alpha3_s(2:end)]);
153                     alpha2_saux=strrep(num2str(alpha2_aux+0.001,'%2.15f'),'.','');
154                     alpha2_saux=alpha2_saux(1:4);
155                     alpha3_saux=strrep(num2str(alpha3_aux+0.001,'%2.15f'),'.','');
156                     alpha3_saux=alpha3_saux(1:4);
157                     Files=dir(['*','_',alpha1_s,'_',alpha2_saux(1:end),'_',alpha3_saux(1:end),'*.inp']);
158                     if ~isempty(Files),
159                         alpha2_s=alpha2_saux;
160                         alpha3_s=alpha3_saux;
161                     else
162                     % + alpha2, - alpha3
163                     alpha2_aux=str2num([alpha2_s(1),'.',alpha2_s(2:end)]);
164                     alpha3_aux=str2num([alpha3_s(1),'.',alpha3_s(2:end)]);
165                     alpha2_saux=strrep(num2str(alpha2_aux+0.001,'%2.15f'),'.','');
166                     alpha2_saux=alpha2_saux(1:4);
167                     alpha3_saux=strrep(num2str(alpha3_aux-0.001,'%2.15f'),'.','');
168                     alpha3_saux=alpha3_saux(1:4);
169                     Files=dir(['*','_',alpha1_s,'_',alpha2_saux(1:end),'_',alpha3_saux(1:end),'*.inp']);
170                     if ~isempty(Files),
171                         alpha2_s=alpha2_saux;
172                         alpha3_s=alpha3_saux;
173                     else
174                     % - alpha2, = alpha3
175                     alpha2_aux=str2num([alpha2_s(1),'.',alpha2_s(2:end)]);
176                     alpha3_aux=str2num([alpha3_s(1),'.',alpha3_s(2:end)]);
177                     alpha2_saux=strrep(num2str(alpha2_aux-0.001,'%2.15f'),'.','');
178                     alpha2_saux=alpha2_saux(1:4);
179                     Files=dir(['*','_',alpha1_s,'_',alpha2_saux(1:end),'_',alpha3_saux(1:end),'*.inp']);
180                     if ~isempty(Files),
181                         alpha2_s=alpha2_saux;
182                         alpha3_s=alpha3_saux;
183                     else
184                     % - alpha2, + alpha3
185                     alpha2_aux=str2num([alpha2_s(1),'.',alpha2_s(2:end)]);
```

```
                    alpha3_aux=str2num([alpha3_s(1),'.',alpha3_s(2:end)]);
                    alpha2_saux=strrep(num2str(alpha2_aux-0.001,'%2.15f'),'.','');
                    alpha2_saux=alpha2_saux(1:4);
                    alpha3_saux=strrep(num2str(alpha3_aux+0.001,'%2.15f'),'.','');
                    alpha3_saux=alpha3_saux(1:4);
                    Files=dir(['*','_',alpha1_s,'_',alpha2_saux(1:end),'_',alpha3_saux(1:end),'*.inp']);
                    if ~isempty(Files),
                        alpha2_s=alpha2_saux;
                        alpha3_s=alpha3_saux;
                    else
                    % - alpha2, - alpha3
                    alpha2_aux=str2num([alpha2_s(1),'.',alpha2_s(2:end)]);
                    alpha3_aux=str2num([alpha3_s(1),'.',alpha3_s(2:end)]);
                    alpha2_saux=strrep(num2str(alpha2_aux-0.001,'%2.15f'),'.','');
                    alpha2_saux=alpha2_saux(1:4);
                    alpha3_saux=strrep(num2str(alpha3_aux-0.001,'%2.15f'),'.','');
                    alpha3_saux=alpha3_saux(1:4);
                    Files=dir(['*','_',alpha1_s,'_',alpha2_saux(1:end),'_',alpha3_saux(1:end),'*.inp']);
                    if ~isempty(Files),
                        alpha2_s=alpha2_saux;
                        alpha3_s=alpha3_saux;
                    end
                    end
                    end
                    end
                    end
                    end
                    end
                    end
                end
                if isempty(Files)
                    new_entrance_inp=[];
                    singular_FE_sidelength_index=[singular_FE_sidelength_index; []];
                    name_index=strvcat(name_index, new_entrance);
                    alpha1_index=[alpha1_index,[]];
                    alpha2_index=[alpha2_index,[]];
                    alpha3_index=[alpha3_index,[]];
                else
                    new_entrance_inp=[id '_' alpha1_s '_' alpha2_s '_' alpha3_s '.inp'];
                    name_index=strvcat(name_index, new_entrance);
                    file_index = strvcat(file_index,new_entrance_inp);
                    singular_FE_sidelength_index=[singular_FE_sidelength_index; singular_FE_sidelength];
                    alpha1_index=[alpha1_index,alpha1];
                    alpha2_index=[alpha2_index,alpha2];
                    alpha3_index=[alpha3_index,alpha3];
                end
            end
        end
    end
end

ncasos=length(file_index);
```

Script 3.25 Step 03 in MATLAB.

3.9.4 step_04_mesher

```
function [coordenadas,coordenadasN5,coordenadas5,triangulos,triangulos5] = step_04_mesher(inp_malla,nofelements,
    sidelength,alpha,typeelement,vct);
xcentro=vct(1); ycentro=vct(2); zcentro=vct(3);

%% Coordinates matrix
begin_reading_coordenadas= strfind(inp_malla,'*Node'); % Reads from "*Node".
end_reading_coordenadas= strfind(inp_malla,'*Element'); % to "*Element".
coordenadas_aux=[inp_malla(begin_reading_coordenadas+8:end_reading_coordenadas-2)]; % The .inp coordinate
    file is reduce to the necessary part, the one corresponding to the nodal coordinates. The +5 is due to read
    the beginning of the numbers, and not the heading. The -2 is bound to not read the following heading
    characters.
```

```
coordenadas_aux= strsplit (coordenadas_aux,'\n'); % The text is divided in lines by the newline character.
coordenadas_aux=coordenadas_aux'; % Vertically arranged text.
for i=1:length(coordenadas_aux)
    coordenadas(i,:) =str2num(cell2mat(coordenadas_aux(i))); % Every row is turned into a text array, and every text
        array (as a cell) is turned into a numeric value.
end
coordenadas(:,1) =[]; % The identification number column is deleted.

%% Elements connectivity matrix

 begin_reading_triangulos = strfind(inp_malla,'*Element,'); % The beginning of the Triangle matrix is found
 end_reading_triangulos = strfind(inp_malla,'*End Part'); % The end of the Triangle matrix is found
 triangulos_aux =[inp_malla( begin_reading_triangulos +21: end_reading_triangulos -2)]; % Their values are introduced in
        a variable
 triangulos_aux = strsplit ( triangulos_aux ,'\n'); % The text is divided in lines by the newline character.
 triangulos_aux =triangulos_aux '; % The matrix is trasposed to get the correct format.
 for i=1:length( triangulos_aux )
     triangulos (i ,:) =str2num(cell2mat( triangulos_aux (i))); % The triangle variable line is turned into numbers
 end
 triangulos (:,1) =[]; % The first column gets deleted since it is unnecessary

%% Singular-Elements Coordinates Matrix

N=2*nofelements; % The number of regular polygon edges whose upper half will be singular elements.
n=(0:nofelements);
r=sidelength*1/2; % alpha=1/2, the position of the node in the singular-behavioured edge.
if typeelement==1 % If there are singular elements.
      x(n+1)=r*cos(2*pi*n/N);
      y(n+1)=r*sin(2*pi*n/N); % Their x and y regular polygon coordinates are calculated
      z(n+1)=zcentro; % Introduce z value.
       coordenadas_singulares =[xcentro, ycentro, zcentro; x', y', z'];
 elseif typeelement==0 % If there are no singular elements
       coordenadas_singulares =[xcentro, ycentro, zcentro];
end
 coordenadas_singulares (:,3) =[]; % The third column gets deleted, since it is unnecessary.

%% Singular elements connectivity matrix.

r=sidelength;
x(n+1)=r*cos(2*pi*n/N);
y(n+1)=r*sin(2*pi*n/N);
z(n+1)=0; % Introduce z coordinate.
for n=1:nofelements+1
       closestpoint (n)=dsearchn(coordenadas,[x(n) y(n)]); % This command finds the closest point (in the mesh) to the
        ones obtained by our formula (they may not perfectly coincide due to minimal Abaqus-MATLAB errors).
end
if typeelement==1 % The connectivity matrix is created for singular elements. The points are ordered clockwise: the
        first node is the central one, the second to its right, and the third one is the pre-existent one (already
       in the mesh, since it is part of both the singular and non-singular element), the fourth point will be the
       other pre-existent node and the last one will be the remaining intermediate point.
      for i=1:nofelements
           triangulos5 (i ,:) =[size(coordenadas,1)+1 size(coordenadas,1)+1+i closestpoint (i) closestpoint (i+1) size(
       coordenadas,1)+2+i];
      end
 elseif typeelement==0
      for i=1:nofelements % In this case, no halfway points exist.
           triangulos5 (i ,:) =[size(coordenadas,1)+1 closestpoint (i) closestpoint (i+1)];
      end
       triangulos =[triangulos ; triangulos5 ];
       triangulos5 =[];
end
%% This line assembles the coordinate matrix.
coordenadas5=coordenadas_singulares;
coordenadasN5=coordenadas;
coordenadas=[coordenadas; coordenadas_singulares ];
```

Script 3.26 Step 04 in MATLAB.

3.9.5 step_05_boundary_conditions

Function step_05_boundary_conditions part 1

```
%% This line finds the plate edges.
[ Lista_nodos_borde_izquierdo , Lista_nodos_borde_superior ,    ...
    Lista_nodos_borde_derecho, Lista_nodos_borde_inferior ,    ...
    Nodos_borde_izquierdo, Nodos_borde_superior,               ...
    Nodos_borde_derecho, Nodos_borde_inferior ],               ...
    =encuentra_bordes(coordenadas, tolerancia );
```

Script 3.27 Step 05 in MATLAB.

Function encuentra_bordes

```
function [ Lista_nodos_borde_izquierdo , Lista_nodos_borde_superior ,Lista_nodos_borde_derecho,
      Lista_nodos_borde_inferior ,Nodos_borde_izquierdo, Nodos_borde_superior,Nodos_borde_derecho,
      Nodos_borde_inferior]=encuentra_bordes(coordenadas, tolerancia )

%% This function generates 8 lists : 4 with the nodal number of each edge nodes (upper, lower, right and left), 4
      with the coordinates of said nodes.

%% Initialize matrices
Nodos_borde_derecho=[]; Lista_nodos_borde_derecho=[];
Nodos_borde_izquierdo=[]; Lista_nodos_borde_izquierdo =[];
Nodos_borde_superior=[]; Lista_nodos_borde_superior =[];
Nodos_borde_inferior =[]; Lista_nodos_borde_inferior =[];

%% Obtain the boundary points from our node list
frontera =boundary(coordenadas(:,1) ,coordenadas (:,2) );

%% Obtain the upper and lower (y coordinate) and most to the right and to the left limits of the coordinates (edges
      )
coordenada_borde_derecho=max(coordenadas(:,1));
coordenada_borde_izquierdo=min(coordenadas(:,1) );
coordenada_borde_superior=max(coordenadas(:,2) );
coordenada_borde_inferior =min(coordenadas(:,2) );

%% We define a tolerance in case there is a mild discordance among the points coordinates
tol=1e-10;

%% Compute borders. We add to each list the nodes whose coordinates fit each limit. If a nodes coordinates
      coincide with the right or left limits (including tolerance), they are added to the list.
for i=1:length( frontera )
    nodo=frontera(i);
    if coordenadas(nodo,1)>=coordenada_borde_derecho-tol && coordenadas(nodo,1)<=coordenada_borde_derecho+tol,
        Nodos_borde_derecho=[Nodos_borde_derecho; coordenadas(nodo,:)];
        Lista_nodos_borde_derecho=[Lista_nodos_borde_derecho; nodo];
    end
    if coordenadas(nodo,1)>=coordenada_borde_izquierdo-tol && coordenadas(nodo,1)<=coordenada_borde_izquierdo+tol,
        Nodos_borde_izquierdo=[Nodos_borde_izquierdo; coordenadas(nodo,:) ];
        Lista_nodos_borde_izquierdo=[ Lista_nodos_borde_izquierdo ; nodo];
    end
    if coordenadas(nodo,2)>=coordenada_borde_superior-tol && coordenadas(nodo,2)<=coordenada_borde_superior+tol,
        Nodos_borde_superior=[Nodos_borde_superior; coordenadas(nodo,:) ];
        Lista_nodos_borde_superior=[ Lista_nodos_borde_superior ; nodo];
    end
    if coordenadas(nodo,2)>=coordenada_borde_inferior-tol && coordenadas(nodo,2)<=coordenada_borde_inferior+tol,
        Nodos_borde_inferior=[Nodos_borde_inferior; coordenadas(nodo,:) ];
        Lista_nodos_borde_inferior =[ Lista_nodos_borde_inferior ; nodo];
    end
end

%% Deletes repeated values with a call to elimina_dobles.
[ Lista_nodos_borde_izquierdo ,Nodos_borde_izquierdo]=elimina_dobles( Lista_nodos_borde_izquierdo ,
      Nodos_borde_izquierdo);
[ Lista_nodos_borde_superior ,Nodos_borde_superior]=elimina_dobles( Lista_nodos_borde_superior ,Nodos_borde_superior);
[Lista_nodos_borde_derecho,Nodos_borde_derecho]=elimina_dobles(Lista_nodos_borde_derecho,Nodos_borde_derecho);
```

```
[ Lista_nodos_borde_inferior ,Nodos_borde_inferior]=elimina_dobles( Lista_nodos_borde_inferior ,Nodos_borde_inferior);
end
```

Script 3.28 Function Encuentra_bordes.

Function elimina_dobles

```
function [Lista_nueva,nodos_nuevos] = elimina_dobles(Lista,nodos)

%% This function deletes repeated adjacent values to make a simple list of boundary nodes.
Lista_nueva=Lista;
nodos_nuevos=nodos; % List of nodes and coordinates copied in a new vector
eliminar =[]; % The pointer vector that will give us the values to erase is set.
for i=1:((length(unique(Lista_nueva)))-1) % If a value coincides with the next value in the array, its position is
      introduced in the vector.
    if Lista_nueva(i)==Lista_nueva(i+1)
        eliminar=[eliminar,i];
    end
end
if Lista_nueva(end)==Lista_nueva(1) % A coincidence between the first and last is also checked.
    eliminar=[eliminar length(Lista_nueva)];
end
Lista_nueva(eliminar)=[];
nodos_nuevos(eliminar,:)=[]; % The repeated values are deleted.
end
```

Script 3.29 Function elimina dobles.

Function step_05_boundary_conditions part 2

```
%% This section orders the edges' nodes clockwise from the lower left node.
 [Lista_nodos,Lista_nodos_borde_izquierdo,...
      Lista_nodos_borde_superior,Lista_nodos_borde_derecho ...
      Lista_nodos_borde_inferior]=                          ...
      ordena_bordes(Lista_nodos_borde_izquierdo,      ...
      Lista_nodos_borde_superior,Lista_nodos_borde_derecho, ...
      Lista_nodos_borde_inferior,Nodos_borde_izquierdo,     ...
      Nodos_borde_superior, Nodos_borde_derecho,            ...
      Nodos_borde_inferior);
```

Script 3.30 Step 05 in MATLAB part 2.

Function ordena_bordes

```
%% The nodes in the vectors must be ordered clockwise, starting with the bottom left node.

function [Lista_nodos,Lista_nodos_borde_izquierdo,...
      Lista_nodos_borde_superior,Lista_nodos_borde_derecho ...
      Lista_nodos_borde_inferior]=ordena_bordes(Lista_nodos_borde_izquierdo, Lista_nodos_borde_superior,...
 Lista_nodos_borde_derecho, Lista_nodos_borde_inferior,Nodos_borde_izquierdo,Nodos_borde_superior,...
 Nodos_borde_derecho,Nodos_borde_inferior)

%% The right Edge is ordered from lower to higher according to their y coordinate. The right-edge nodes are
      introduced in said order, and then is flipped.
[nodos_sorted,nodos_orden]=sort(Nodos_borde_derecho(:,2));
Lista_nodos_borde_derecho=Lista_nodos_borde_derecho(nodos_orden,:);
Lista_nodos_borde_derecho= flip(Lista_nodos_borde_derecho);
%% The left Edge does not require the flip.
[nodos_sorted,nodos_orden]=sort(Nodos_borde_izquierdo(:,2));
Lista_nodos_borde_izquierdo=Lista_nodos_borde_izquierdo(nodos_orden,:);
%% Idem for the top Edge nodes from left to right.
[nodos_sorted,nodos_orden]=sort(Nodos_borde_superior(:,1));
Lista_nodos_borde_superior=Lista_nodos_borde_superior(nodos_orden,:);
%% Finally the lower Edge receives the same treatment, but is flipped afterwards.
```

```
[nodos_sorted,nodos_orden]=sort(Nodos_borde_inferior(:,1));
Lista_nodos_borde_inferior = Lista_nodos_borde_inferior(nodos_orden,:);
Lista_nodos_borde_inferior = flip( Lista_nodos_borde_inferior );

%% This section generates a list with all the nodes ordered, and the double values are deleted.
Lista_nodos=[Lista_nodos_borde_izquierdo; Lista_nodos_borde_superior; ...
             Lista_nodos_borde_derecho; Lista_nodos_borde_inferior ];
Nodos=[Nodos_borde_izquierdo; Nodos_borde_superior; Nodos_borde_derecho; ...
             Nodos_borde_inferior ];
[Lista_nodos,Nodos]=elimina_dobles(Lista_nodos,Nodos);
end
```

Script 3.31 Function Ordena Bordes.

Function step_05_boundary_conditions part 3

```
%% For different boundary conditions, fill their definition here.
%% Values must be introduced always clockwise. In our case: there are no nodes with Dirichlet boundary condition.
    Neumann nodes are top edge (value 1), lateral and bottom edge left half (value 0). Robin nodes are bottom
    edge right half (parameter k). Except for the singular element nodes, which are bottom central and the
    following two sideways.
nodos_dirichlet =[];
nodos_neumann=[Lista_nodos_borde_izquierdo;...
    Lista_nodos_borde_superior ;...
    Lista_nodos_borde_derecho ;...
    Lista_nodos_borde_inferior ( floor(length( Lista_nodos_borde_inferior )/2)+3:end) ;...
    Lista_nodos_borde_izquierdo(1) ];

%% We select the left, upper, right edges, and from the bottom edge, from the rightmost to the midway but three
    node. The left edge first node is added again to close the circle (between the last and first nodes).
if rem(length( Lista_nodos_borde_inferior ),2)==0
    % Si el número de nodos en el borde inferior fuera par, los nodos con condición Robin irían desde el primero
    hasta la mitad menos dos nodos.
%% If the number of nodes in the bottom edge is even, Robin nodes span from the the first to the midway but two
    nodes.
nodos_robin=[ Lista_nodos_borde_inferior (1:( length( Lista_nodos_borde_inferior )/2)-2)];
else
%% If the number is odd, the formula is slightly different: the downward nearest whole number, and subtract one to
    arrive to the desired node.
nodos_robin=[ Lista_nodos_borde_inferior (1: floor(length( Lista_nodos_borde_inferior )/2)-1)];
end

%% No singular nodes affected by Dirichlet condition.
nodos_dirichlet5 =[];
%% We apply the same reasoning to non-singular elements. In this case, Neumann nodes span from midway edge node to
    the following two left nodes.
if rem(length( Lista_nodos_borde_inferior ),2)==0
    nodos_neumann5=[Lista_nodos_borde_inferior((length( Lista_nodos_borde_inferior )/2) :( length(
        Lista_nodos_borde_inferior )/2)+2) '];
else
%% And again, we must differentiate if the number of elements in the bottom edge is even or odd.
nodos_neumann5=[Lista_nodos_borde_inferior( floor(length( Lista_nodos_borde_inferior )/2) : floor(length(
        Lista_nodos_borde_inferior )/2)+2) '];
end
nodos_robin5=[ Lista_nodos_borde_inferior ( floor(length( Lista_nodos_borde_inferior )/2) :-1: floor(length(
        Lista_nodos_borde_inferior )/2)-2) '];

%% The nodes are again rearranged according to the nodal list in case they had changed during the process. In
    general, they may not maintain its order.
if ~isempty( nodos_dirichlet )
    nodos_dirichlet =reordena( nodos_dirichlet ,Lista_nodos);
else dirichlet =[];
end
if ~isempty(nodos_neumann)
    nodos_neumann=reordena(nodos_neumann,Lista_nodos);
else neumann=[];
end
if ~isempty(nodos_robin)
    nodos_robin=reordena(nodos_robin,Lista_nodos);
```

```
else  robin =[];
end
if  ~isempty( nodos_dirichlet5 )
      nodos_dirichlet5 =reordena( nodos_dirichlet5 ,Lista_nodos);
else   dirichlet5  =[];
end
if  ~isempty(nodos_neumann5)
     nodos_neumann5=reordena(nodos_neumann5,Lista_nodos);
else  neumann5=[];
end
if  ~isempty(nodos_robin5)
     nodos_robin5=reordena(nodos_robin5,Lista_nodos);
else  nodos_robin5=[];
end
```

Script 3.32 Step 05 in MATLAB part 3.

Function reordena

```
function  nodos_reordenados = reordena(nodos,Lista_nodos)

%% The list  order  is  changed  if  the  list  is  circular  to  close  it  properly .
if  ( intersect (nodos,Lista_nodos(1))~=0 &  intersect (nodos,Lista_nodos(end))~=0)==1
     Lista_nodos(end+1)=Lista_nodos(1);
     Lista_nodos(1)=0;
end

%% Repeated nodes are deleted  from the  list .
nodos_2=unique(nodos);
s=1;
for  i=1:length(Lista_nodos)
     if   intersect (Lista_nodos(i),nodos_2)~=0
          nodos_reordenados(s)=Lista_nodos(i);
          s=s+1;
     end
end

%% Each node in the  list  must be  in  the  non-repeated node  list .  If  it  is  in  the  list ,  its  value  is  stored  in  the  re
      -ordered  list .
End
```

Script 3.33 Function Reordena.

Function step_05_boundary_conditions part 4

```
%% Nodal values. Our case  study  values  are  introduced  for  our  case .  More precisely ,  Neumann values are  introduced
      by  parts :  a  zeroes  vector  with  the  same number of components as the  left  edge,  a  ones  vector  with  the  number
      of components of the  top  edge,  another  zeroes  vector  with  the  right  edge number of components, and a  zeroes
      vector  with  the  non-singular  elements  in  the  left  edge  (the  crack ).
%% The sizes of  the  vectors  must be  slightly  changed to  avoid  overlapping :  the  first  and second vector  will  be
      reduced by  one  position .
 valores_dirichlet  =[];
valores_neumann=[0*ones(1,length( Lista_nodos_borde_izquierdo )-1) ...
     ones(1, length ( Lista_nodos_borde_superior )-1) ...
     0*ones(1, length (Lista_nodos_borde_derecho)) ...
     0*ones(1, length ( Lista_nodos_borde_inferior ( floor ( length ( Lista_nodos_borde_inferior ) /2)+3:end))-1)];
 valores_dirichlet5  =[];
valores_neumann5=0.*ones(1,length(nodos_neumann5));

%% Dirichlet  values .  Edge-connectivity  matrices  must be  generated  and the  values  in  the  corresponding  arista .  It
      must have  the  following  form:  [one node,  the  following  node  to  which the node is  connected,  value  of  the  load
       in  the  first  node applied  to  the  arista  between nodes,  value  of  the  load  in  the  second node applied  to  the
      arista  between nodes ].  A new line  is  applied  to  link  the  first  and  last  node. Because of the process ,  there
      will  be  aristae  between unlinked nodes. To delete  those  between non-consecutive  aristae ,  the  function  '
      desconectar '  is  defined .
if  ~isempty( nodos_dirichlet )
```

```
13      conexion_dirichlet =[ nodos_dirichlet (1:end-1)'  nodos_dirichlet (2:end) ';  nodos_dirichlet (end)  nodos_dirichlet (1)
          ];
14      desplazamientos_dirichlet =[ valores_dirichlet (1:end-1)'  valores_dirichlet (2:end) ';  valores_dirichlet (end)
          valores_dirichlet (1) ];
15      dirichlet =[ conexion_dirichlet  desplazamientos_dirichlet ];
16      dirichlet =desconectar( dirichlet ,Lista_nodos);
17  else  dirichlet =[];
18  end
```

Script 3.34 Step 05 in MATLAB part 4.

Function desconectar

```
1  function condicion_nueva=desconectar(condicion,Lista_nodos)
2   lista_eliminacion =[];
3  s=1;
4      for i=1:size(condicion,1)
5          pos=find(Lista_nodos==condicion(i,1)); % Encontramos dónde la lista de nodos coincide con el primer nodo
         que quieres desconectar. We find out where the node list coincides with the first node to disconnect.
6          if pos==1 % If it is the first in the node list
7              if condicion(i,2)~=Lista_nodos(pos+1) && condicion(i,2)~=Lista_nodos(end) % If the following term is
         neither the following in the list or the last one, we store the number of that line in the delete list.
8                  lista_eliminacion (s)=i;
9                  s=s+1;
10                 end
11         end
12         if pos==length(Lista_nodos) % If it is the last term, we do the same with the last-but-one and first term.
13             if condicion(i,2)~=Lista_nodos(1) && condicion(i,2)~=Lista_nodos(pos-1)
14              lista_eliminacion (s)=i;
15             s=s+1;
16             end
17         end
18         if pos~=1 && pos~=length(Lista_nodos) % For intermediate cases
19         if condicion(i,2)~=Lista_nodos(pos+1) && condicion(i,2)~=Lista_nodos(pos-1) % If the following term in the
         connectivity matrix does not belong in the list with the following or previous term.
20             lista_eliminacion (s)=i; % We store that lines number in the delete list.
21             s=s+1;
22         end
23         end
24     end
25     condicion( lista_eliminacion ,:) =[]; % And we proceed to delete said row in the connectivity matrix.
26      lista_eliminacion =[];
27     condicion_nueva=condicion;
28 end
```

Script 3.35 Function desconectar.

Function Step_05_boundary_conditions part 5

```
1  %% Neumann. Same process.
2  if ~isempty(nodos_neumann)
3      conexion_neumann=[nodos_neumann(1:end-1)' nodos_neumann(2:end)'; nodos_neumann(end)' nodos_neumann(1)'];
4      tensiones_neumann=[valores_neumann(1:end-1)' valores_neumann(2:end) ';  valores_neumann(end) valores_neumann(1)];
5      neumann=[conexion_neumann tensiones_neumann];
6      neumann=desconectar(neumann,Lista_nodos);
7  else  neumann=[];
8  end
9
10 %% Robin. Same process.
11 if ~isempty(nodos_robin)
12     conexion_robin=[nodos_robin(1:end-1)'  nodos_robin(2:end) ';  nodos_robin(end)  nodos_robin(1) ];
13     robin=[conexion_robin ];
14     robin=desconectar(robin,Lista_nodos);
15 else  robin =[];
16 end
17
```

```
%% Special Dirichlet. For special triangles, its aristae are defined by three points with the following formula: [a node, the following node to which the node is connected, the following node, load value on the first node applied in the aristae between both nodes, load value on the second node, load value on the third node].
if ~isempty( nodos_dirichlet5 )
    conexion_dirichlet5 =[ nodos_dirichlet5 (1:end-2) nodos_dirichlet5 (2:end-1) nodos_dirichlet (3:end) ];
    desplazamientos_dirichlet5 =[ valores_dirichlet5 (1:end-2)' valores_dirichlet5 (2:end-1)' valores_dirichlet (3:end) ];
    dirichlet5 =[ conexion_dirichlet5 desplazamientos_dirichlet5 ];
    dirichlet5 =desconectar( dirichlet5 ,Lista_nodos);
else dirichlet5 =[];
end
end

%% Special Neumann
if ~isempty(nodos_neumann5)
    conexion_neumann5=[nodos_neumann5(1:end-2) nodos_neumann5(2:end-1) nodos_neumann5(3:end)];
    tensiones_neumann5=[valores_neumann5(1:end-2)' valores_neumann5(2:end-1)' valores_neumann5(3:end) '];
    neumann5=[conexion_neumann5 tensiones_neumann5];
    neumann5=desconectar(neumann5,Lista_nodos);
else neumann5=[];
end

%% Special Robin
if ~isempty(nodos_robin5)
    conexion_robin5=[nodos_robin5(3:end)' nodos_robin5(2:end-1)' nodos_robin5(1:end-2) '];
    robin5=[conexion_robin5 ];
    robin5=desconectar(robin5 ,Lista_nodos);
else nodos_robin5=[];
end

%% Artificial Matrices. When there are leaps on the load values, for example, the load value change around the corners from 0 to 1, the result of this process will assign a 1 or 0 value to the node position independently from the arista. Hence, to define aristae with different values, we must change the nodal value at the arista definitions (some nodes will have a different value depending on the arista).

neumann(length(Lista_nodos_borde_izquierdo)-1,3)=1;
neumann(length(Lista_nodos_borde_izquierdo)+length(Lista_nodos_borde_superior)-2,3)=0;
nodos_calculo= Lista_nodos_borde_inferior ( ceil (end/2));
```

Script 3.36 Step 05 in MATLAB part 5.

3.9.6 Function step_13_numerical_regressions

The in-between steps are discussed in [17] and have not been changed.
The numerical regressions allows us to analyze the data. Hence, we need to represent our results in a certain manner to figure out the best meshing option. The criteria and data arrangement for each case is discussed in the following code.

```
function [ ajuste , ajuste_2 , slope_index ,max_2nd_crown_change,casos_buenos,nota_a,nota_b, criterio_1 , criterio_2 , criterio_3a , criterio_3b , criterio_4 ] = step_13_numerical_regressions (alpha1_index , alpha2_index , alpha3_index , desplazamientos_calculo , coordenadas_calculo , tipo1 , tipo2 , id , separadores , coordenadas, rcts )
cd([ 'C:\Users\Usuario\Desktop\TFG (Avances y Bibliografia )\Macros b\' , id (1: separadores (1)-1), '\' , id ]);
% clc

%% X: alpha_2, Y: alpha_3, Z: u_ct; X coordinate: alpha_2; Y coordinate: alpha_3; Z coordinate: u_ct

%% Variables initialization
leyenda =[]; ajuste =[];
x_pr=[]; y_pr=[]; z_pr=[];
casos_buenos=[]; slope_index =[]; level_curve =[]; point_vector =[]; max_2nd_crown_change=[];
change=[1]; change2=[1];

%% This section fills sparce matrices (FEM results).
desplazamientos_calculo = full ( desplazamientos_calculo );
coordenadas_calculo= full (coordenadas_calculo);

%% This section finds where the indices change, that is, all the cases with the same alpha_1 value.
for s=1:length (alpha1_index)-1
```

```
    if alpha1_index(s)-alpha1_index(s+1)~=0 % Each value that does not coincide with the following
        change=[change, s+1]; %       its position is added as the first unequal values .
    end
end

%% For the one-crown cases:
if tipo2==0

    %% Finding an optimal point. We must find the right proportion between alpha_1 and alpha_2 that results in an
      u_ct value of 2.25596. To do that, we plot u_ct vs alpha_2 for each alpha_1 (there will be several curves).
      If we also plot u_ct=2.25596 we can find a cut-off point between curves (since they are defined by points, we
       will be using a spline to find the cut-off point). For each alpha_1 value we find the alpha_2 and alpha_3
      values that gives the right u_ct. A simple numerical correlation is presupposed.
    % For one-crown cases, we can scatter alpha_1 vs optimal alpha_2 values. Then, a ployfit regression is made.

    for s2=1:length(change)-1 % For each curve (there are as many curves as changes in vector alpha_1) taken as an
      entry of the vector change

        %%    The crack tip displacements are plotted
        alpha1_legend=num2str(alpha1_index(change(s2)));
        % First figure: u_ct vs alpha_2 for each alpha_1
        figure(1)
        xlabel('\alpha_2'); ylabel('u_1');
        plot(alpha2_index(change(s2):change(s2+1)-1), desplazamientos_calculo(change(s2):change(s2+1)-1,1), '
      linewidth',2)
        if s2~=length(change)-1
            leyenda=strvcat(leyenda,['\alpha_1=',alpha1_legend]);
        end
        hold on

        %% This section calculates the intersection between u_ct and alpha_1. A spline is generated with the
      previously calculated points (the FEM results).
        y_dado=desplazamientos_calculo(change(s2):change(s2+1)-1,1); % displacement values
        x_dado=alpha2_index(change(s2):change(s2+1)-1); % alpha_2 values (corresponding to the values between the í
      ndices of alpha_1 changes).
        desiredy = 2.25596; % Obtained by a very precise calculation with fine meshes, cited before
        fun = @(x) desiredy - spline(x_dado,y_dado,x); % We must find a zero of this function: the spline value
      must be equal to the desired value.
        x=sym('x','positive');
        desiredx = fzero(fun, 0.5);
        Interseccion_alpha2(s2) = desiredx;
        longitud_el_sing(s2) = (alpha1_index(change(s2)));
        longitud_el_cor(s2) = (desiredx-alpha1_index(change(s2)));


        %% Slope calculation. The slopes at the cut-off point can be used as mesh quality index as explained in the
       following section.
        for s3=1:length(x_dado)-1; % Among the values of alpha_2 from the curve
            if (desiredx>x_dado(s3) && x_dado(s3+1)>desiredx), % we find the values between which the cutt-off is.
                point_vector=[point_vector; desiredx, desiredy];
                index=s3;
                casos_buenos=[casos_buenos; alpha1_index(change(s2))]; % We only consider the case as valid if the
      cut-off is found by interpolation instead of extrapolation. Out of the computed values, in general there is
      no geometrically feasible crown: alpha_2 would be either smaller than alpha_1 or bigger than the half-
      hexagon.
                slope=(y_dado(index+1)-y_dado(index))/(x_dado(index+1)-x_dado(index)); % From the previous data,
      the segments tangent slope is calculated and introduce in an index.
                slope_index=[slope_index;slope];
            end
        end
    end
    scatter(point_vector(:,1), point_vector(:,2));

    %% Adjustment (linear regression).
    [ajuste,gofa]=fit(longitud_el_sing(floor(casos_buenos*20))', longitud_el_cor(floor(casos_buenos*20))', 'poly1');
      % A first adjustment only with the good cases
    [ajuste_2,gofb]=fit(longitud_el_sing(floor(casos_buenos*20))', longitud_el_cor(floor(casos_buenos*20))', 'poly2')
      ; % And a different one with all the cases.
    s2=s2+1;
    %% Plot curves u_ct vs alpha_1
    plot(alpha2_index(change(s2):end), desplazamientos_calculo(change(s2):end,1), 'linewidth',2)
```

```
74      leyenda= strvcat (leyenda ,+[ '\alpha_1=' ,num2str(alpha1_index(end)) ]) ;
75      legend(leyenda)
76      xlabel ('\alpha_2')
77      ylabel ('u_1')
78      xlim([0.05   1.1])
79      hold  on
80      plot  ([0:1],[2.25596,2.25596])
81      print ('-dpng',  '-r300',[ 'graph_curvas_uvsalpha2' , id , '.png' ])
82
83      %% Adjustment curve plot
84      % Second figure:  scattered  points  vs  1st  degree  adjustment
85      figure (2);  xlabel ('\alpha_1'),  ylabel ('\alpha_2-\alpha_1')
86      ajuste_lin = polyfit ( longitud_el_sing ( floor (casos_buenos*20)), longitud_el_cor ( floor (casos_buenos*20)),1) ;
87      scatter ( longitud_el_sing ( floor (casos_buenos*20)), longitud_el_cor ( floor (casos_buenos*20))) ;  hold  on;
88      x_ajuste_lin =[ longitud_el_sing ( floor (casos_buenos(1)*20)) :0.01:  longitud_el_sing ( floor (casos_buenos(end)*20)) ];
89      y_ajuste_lin = ajuste_lin (1) .* x_ajuste_lin + ajuste_lin (2) ;
90      plot ( x_ajuste_lin , y_ajuste_lin , 'Linewidth' ,2) ;  hold  on;
91      print ('-dpng',  '-r300',[ ' graph_curvas_ajuste_ ' , id , '.png' ])
92      % Third  figure :  scattered  points  vs  2nd degree  adjustment
93      figure (3)
94      ajuste_par = polyfit ( longitud_el_sing ( floor (casos_buenos*20)), longitud_el_cor ( floor (casos_buenos*20)),2) ;
95      scatter ( longitud_el_sing ( floor (casos_buenos*20)), longitud_el_cor ( floor (casos_buenos*20))) ;  hold  on;
96      x_ajuste_par =[ longitud_el_sing (round(casos_buenos(1)*20)) :0.01:  longitud_el_sing (round(casos_buenos(end)*20)) ];
97      y_ajuste_par = ajuste_par (1) .* x_ajuste_par .^2+ ajuste_par (2) .* x_ajuste_par + ajuste_par (3) ;
98      plot ( x_ajuste_par , y_ajuste_par ) ;  hold  on;
99      print ('-dpng',  '-r300',[ ' graph_curvas_ajuste_todos_ ' , id , '.png' ])
100
101     %% Score
102     if  isempty(casos_buenos)==0
103          criterio_1  = length (casos_buenos)/ length (alpha1_index (change)) ;
104          theta  = atan(mean(slope_index)) ;
105          criterio_2  = (pi/2+ theta ) /( pi /2) ;
106         x = longitud_el_sing ( floor (casos_buenos*20));      % Create     x
107         y = longitud_el_cor ( floor (casos_buenos*20));       % Create     y
108         Rsqa = gofa . rsquare ;  % Goodness of  fit  for  the  linear  regression
109         Rsqb = gofb . rsquare ;  % Gooodness of  fit  for  the  parabolic  regression
110          criterio_3a =Rsqa;  criterio_3b =Rsqb;
111          criterio_4 =1-size(coordenadas ,1) /1131;
112         nota_a=( criterio_1 + criterio_2 + criterio_3a + criterio_4 ) /4;
113         nota_b=( criterio_1 + criterio_2 + criterio_3b + criterio_4 ) /4;
114     else
115          criterio_1 =0;  criterio_2 =0;  criterio_3a =0;  criterio_3b =0;  criterio_4 =0;
116         nota_a=0;  nota_b=0;
117     end
118 end
119
120 %% For two-crown cases
121 if  tipo2~=0
122
123     %% This section  finds  the  optimal  curve . Now there  are  three  axis: alpha_2, alpha_3 and u_ct . The surface  that
          relates  those  parameters  for  every  alpha_1 . When that  surface  is  cut  by a plane  with  constant  value  u_ct
         =2.25596 we obtain  a  level  curve from which we can  extract  to  values . The  first  one, the  variation  in the
          solution  produced by the  second crown. The second one, a  relationship  among optimal values  for  alpha_1,
         alpha_2  and alpha_3  (as  an  adjustment  plane) .
124     for  s2=1: length (change)-1
125          desiredz  = 2.25596;
126         alpha1_pr=alpha1_index(change(s2)) ;
127
128         %% This section  interpolates  the  surface  points .
129         X=alpha2_index(change(s2):change(s2+1)-1) ';
130         Y=alpha3_index(change(s2):change(s2+1)-1) ';
131         Z=desplazamientos_calculo(change(s2) :change(s2+1)-1,1) ';
132         x = X; y = Y; z = Z;
133
134         %% This section  identifies  if  the  case  is  valid  or  not . Since the  surface  is  continuous, if  its  maximum
          value is superior  to  the  desired  one and its  minimal value  is  below the  desired  value, the  surface  will  acquire
           the  desired  value  at  some point .
135          if  max(Z(:))> desiredz  && min(Z(:))<desiredz
136              casos_buenos=[casos_buenos;  alpha1_index(change(s2)) ];
137          end
138
```

```
%% Surface points are also interpolated.
bound=boundary(x,y); % This line finds the contour of the Surface projected on the z=cte plane.
for s3=1:length(Y) % From the previous tests we have observed that the level curve y vs x is biyective, but x vs y is not. For this process, we must do as follows: we generate a spline with sparce values from the previously calculated meshes (u_ct, alpha_2, alpha_3). For each value of alpha_3 (y) from the index, we use the interpolating function maintaining the y value constant while keeping x_interp as a symbolic variable (representing alpha_2). The x cut-off value of our surface with the u_ct-constant plane for the fixed y is found by the function fzero.
y_interp=Y(s3);
F = scatteredInterpolant(X,Y,Z');
fun = @(x_interp) F(x_interp,y_interp) - desiredz;
x_interp=sym('x_interp','positive');
desiredx = fzero(fun, 0.5);
if inpolygon(desiredx,y_interp,x(bound),y(bound))==1, % The following code will only be executed if the cut-off point is inside the proyected surface, otherwise the case will not be valid since the relationship between alpha_1, alpha_2 and alpha_3 is not allowed (due to geometric limitations).
level_curve=[level_curve; desiredx, y_interp, desiredz]; % If it is valid, this point is added to the level curve. This vector will define the x, y, z coordinates of the intersection curve.
%%
x_dado=alpha2_index(change(s2):change(s2+1)-1);
y_dado=desplazamientos_calculo(change(s2):change(s2+1)-1,1);
for s3=1:length(x_dado)-1; % Same process as before
if (desiredx>x_dado(s3) && x_dado(s3+1)>desiredx),
index=s3;
slope=(y_dado(index+1)-y_dado(index))/(x_dado(index+1)-x_dado(index));
slope_index=[slope_index;slope];
end
end
% First figure: level curve points and 3D curve of results
% (that should be contained within the surface).
figure(1)
hold on
scatter3(level_curve(:,1),level_curve(:,2),level_curve(:,3),'Linewidth',3) % Curve points are plotted.
end
end
hold on
plot3(X,Y,Z) % A 3D-curve is plotted by the points resulting from simulations.
hold on

%% Level curve adjustment (with the axis changed to fit a parabolic distribution).
if isempty(level_curve)==0 % If the level curve exists
ajuste_entry = polyfit(level_curve(:,2),level_curve(:,1),2); % The level curve data are adjusted.
ajuste=[ajuste; ajuste_entry];
x_pr=[x_pr; level_curve(:,1)]; % Values are introduced
y_pr=[y_pr; level_curve(:,2)];
% More precisely: for each alpha_1, the relationship between alpha_2 y alpha_3 that produces the right u_ct is not very dependant on alpha_3. Said parabolas (that produced the adjustment) are very close to straight lines. For a fixed alpha_2 value, the difference between maximum and minimum u_ct values (that depend only on alpha_3) is stored.
z_pr=[z_pr; alpha1_pr*ones(size(level_curve(:,1)))]; % The maximal variation produced by the second crown is stored.
level_curve=[];
end

%% Maximal curve variation
alpha2_alpha1_cte=[]; uct_alpha_1_cte=[]; change2=[];
alpha2_alpha1_cte=alpha2_index(change(s2):change(s2+1));
uct_alpha_1_cte=desplazamientos_calculo(change(s2):change(s2+1));
for s3=1:length(alpha2_alpha1_cte)-1
if alpha2_alpha1_cte(s3)-alpha2_alpha1_cte(s3+1)~=0 % Same selection process
change2=[change2, s3+1];
end
end
for s4=1:length(change2)-1, % For each and every repeated alpha_2 value, the subtraction between maximum and minimum u_ct values is stored.
alpha2_cte=alpha2_alpha1_cte(change2(s4):change2(s4+1)-1);
uct_alpha_2_cte=uct_alpha_1_cte(change2(s4):change2(s4+1)-1);
max_2nd_crown_change=[max_2nd_crown_change;abs(max(uct_alpha_2_cte)-min(uct_alpha_2_cte))];
end
```

```
196             %% Curve plot
197             xv = linspace(min(x), max(x), 20); % Linear alpha_2 interpolation values
198             yv = linspace(min(y), max(y), 20); % Linear alpha_3 interpolation values
199             [X,Y] = meshgrid(xv, yv); % X-Y reticule generation
200             Z = griddata(x,y,z,X,Y); % Sparce data interpolation to fit the Surface.
201             surf(X,Y,Z)
202         end
203
204         %% Ajuste de los planos
205         Xcolv = x_pr(:); % alpha_2 from the level curve
206         Ycolv = y_pr(:); % alpha_3 from the level curve
207         Zcolv = z_pr(:); % alpha_1 from the alpha_1 index
208         Const = ones(size(Xcolv));
209         Coefficients = [Xcolv Ycolv Const]\Zcolv; % This section computes the plane coefficients by the Least Squares
              Method.
210         XCoeff = Coefficients(1);
211         YCoeff = Coefficients(2);
212         CCoeff = Coefficients(3);
213
214         %% alpha_1-alpha_2-alpha_3 adjustment.
215         % Third figure: fitting planes (alpha_1 vs alpha_3 vs alpha_3)
216         figure(2)
217         L=plot3(x_pr,y_pr,z_pr,'ro'); % Plot the original data points
218         set(L,'Markersize',2*get(L,'Markersize')) % Making the circle markers larger
219         set(L,'Markerfacecolor','r') % Filling in the markers
220         hold on
221         [xx, yy]=meshgrid(0:0.1:1,0:0.1:1); % Generating a regular grid for plotting
222         zz = XCoeff * xx + YCoeff * yy + CCoeff;
223         surf(xx,yy,zz) % Plotting the surface
224         title(sprintf('z=(%f)*x+(%f)*y+(%f)',XCoeff, YCoeff, CCoeff))
225         xlabel('\alpha_2'); ylabel('\alpha_3'); zlabel('\alpha_1');
226         print('-dpng', '-r300', ['graph_plano_ajuste_',id,'_.png'])
227         ajuste_2=[-XCoeff -YCoeff 1 CCoeff]; %% Z coordinate pointing upwards
228
229         %% MATLAB Surface fit
230         Coeff_vectora=[0 0]; Rsquare_plane_a=0;
231         Coeff_vectorb=[0 0 0]; Rsquare_plane_b=0;
232         if size(x_pr,1)>=3
233             [Coeff_vectora,gofa]= fit([x_pr, y_pr],z_pr,'poly11');
234             Rsquare_plane_a=gofa.rsquare; % Goodness of fit for the linear regression
235         end
236         if size(x_pr,1)>6
237             [Coeff_vectorb,gofb]= fit([x_pr, y_pr],z_pr,'poly22');
238             Rsquare_plane_b=gofb.rsquare; % Goodness of fit for the parabolic regression
239         end
240
241         %% Right-solution-plane graphic
242         % First and second figures: surfaces (u_ct vs alpha_2 vs alpha_3 for each alpha_1)
243         figure(1); grid on; xlabel('\alpha_2'); zlabel('u_1'); ylabel('\alpha_3'); hold on
244         [X,Y] = meshgrid (0:0.7:0.7,0.1:0.8:0.9) ;
245         Z = 2.25596*ones(size(X));
246         surf(X,Y,Z); view(45,20); axis([0 0.8 0 1 1.75 2.5]);
247         print('-dpng', '-r300', ['graph_curva_ajuste_',id,'.png']);
248         view([0 0 1]); print('-dpng', '-r300', ['graph_curva_ajuste_arriba_',id,'.png'])
249
250         %% Score criteria
251         if isempty(casos_buenos)==0
252             criterio_1 = length(casos_buenos)/length(alpha1_index(change));
253             theta = atan(mean(slope_index));
254             criterio_2 = (pi/2+theta)/(pi/2);
255             criterio_3a = Rsquare_plane_a;
256             criterio_3b = Rsquare_plane_b;
257             criterio_4 =1-size(coordenadas,1)/1131;
258             nota_a=(criterio_1 + criterio_2 + criterio_3a + criterio_4)/4;
259             nota_b=(criterio_1 + criterio_2 + criterio_3b + criterio_4)/4;
260         end
261         if isempty(casos_buenos)==1
262             criterio_1 =0; criterio_2 =0; criterio_3a =0; criterio_3b =0; criterio_4 =0;
263             nota_a=0; nota_b=0;
264         end
265     end
```

```
[nota_a  nota_b];
close  all
cd([ 'C:\Users\Usuario\Desktop\TFG (Avances y  Bibliografia )\Macros b\Codigo Matlab' ]);
```

Script 3.37 Step 13 in MATLAB.

3.10 Results

Once the meshes are obtained and simulated, the results must be analyzed in some manner. First, let us plot the results to have a quick overview.

One-crown cases: In each graphic, the number of singular elements and the crown type remains constant, there is one curve for every α_1 value, which relates the crack tip displacement (u_{ct}) with α_2.

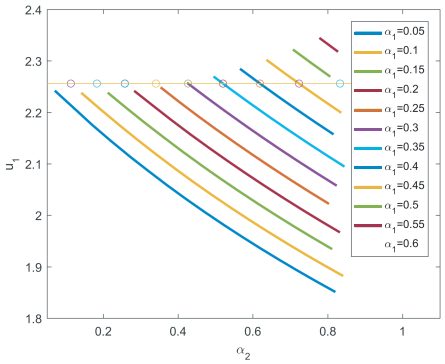


Figure 3.32 One possible typology: few intersections. Cut-off points are extrapolations (not included in the domain of the curve)..

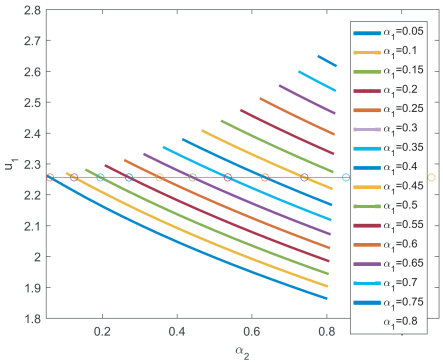


Figure 3.33 Another possible typology: several intersections..

Cut-off points of each curve with the horizontal segment (which is the exact solution) will give us the optimal proportion between radii $alpha_1$ and $alpha_2$. And indirectly, the optimal proportion between singular elements and crown elements. So when the crown element size is plot versus the singular element given by the real solution, we obtain points that can be fit to a curve.

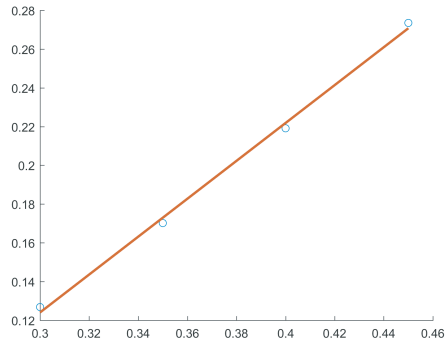


Figure 3.34 With few instances, a fitting curve gives a good approximation.

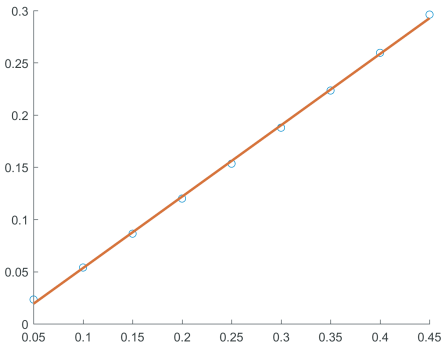


Figure 3.35 With more instances, the best-fitting curve resembles a parabola. Both curves will be used: the line for simplicity and the parabola for accuracy..

Two-crown cases: in each graphic the number of singular elements and the crown types remains constant, and for each α_1 there will be a different surface, which relates u_{ct} to α_2 in one axis and α_3 in the other axis. Since linear and parabolic adjustments are so close to each other, we will present parabolic adjustment graphics from now on.

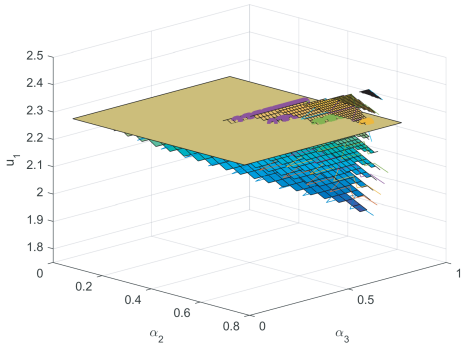


Figure 3.36 Typology: few intersections..

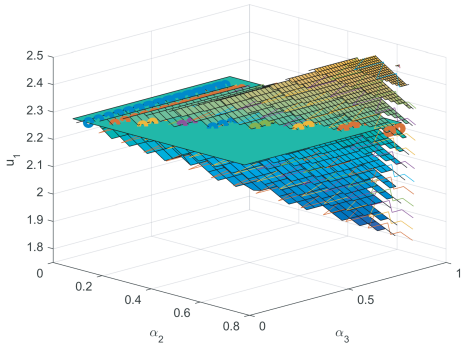


Figure 3.37 Typology: several intersections..

The 3D surfaces resemble an extruded version of the 2D curves, that is, their projection into an $\alpha_2 - \alpha_1$ plane follow the same shape as the 2D cases. That is to say, the projections follow approximately the shape of the one-crown case curves. That indicates that the result has little dependency on the second crown, since independently from its size, the obtained result does not change considerably. The second crown effect is only a slight warp that turns the u_{ct}-level curves into parabolas bended towards the decreasing α_2 direction (as shown in the image). This variation due to the third crown is measured by the code.

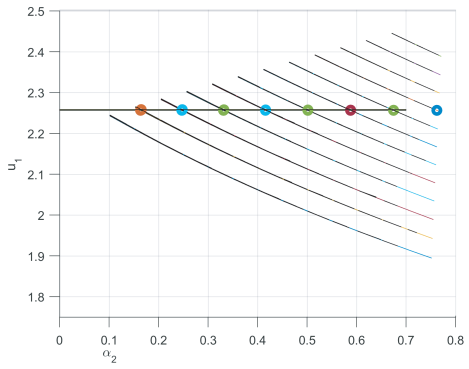


Figure 3.38 Their projection resembles the previous result.

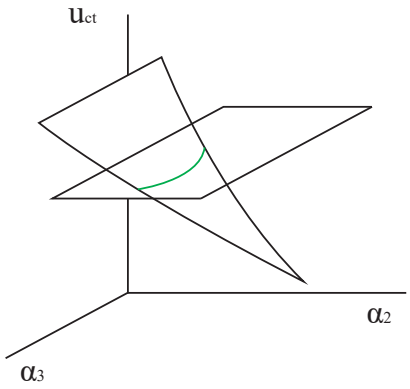


Figure 3.39 Parabolic behavior induced by the second crown (effect in green).

Now, the points that result in the proper result have three coordinates: α_1, α_2, α_3. When scattered they appear to have a planar distribution:
As we can see, the results have a good fit with a plane whereas the one-crown cases have a better fit with a parabola. By now, we can appreciate that the two-crowns results fit adequately enough in a plane whereas the one-crown cases strongly fit in a parabola. However, it is worth noting that the points are not perfectly aligned in a plane either: they have a parabolic tendency although less accentuated than one-crown cases. Projecting the points onto the $\alpha_2 - \alpha_1$ plane allows us to spot said a parabolic-like tendency.
We can use this code to find out a first approximation to what the best proportion between the sizes of the singular elements rosette and the first crown. Only the linear (for one crown) and planar (for two crowns) will be shown, as they are more easily understandable. The parabolic and quadratic-surface adjustment (just as their optimal proportion) will be shown and analysed only for a set of cases (a set of good configurations)

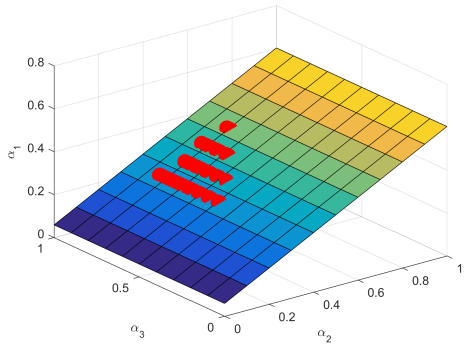


Figure 3.40 Typology: few intersections..

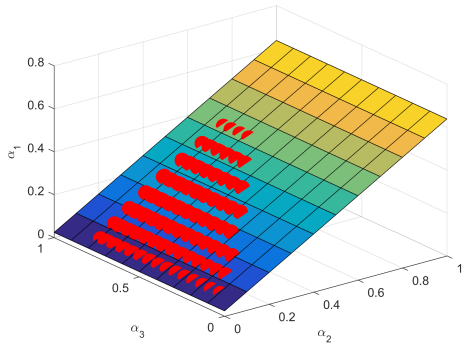


Figure 3.41 Typology: several intersections..

which will be discussed later. For one-crown cases, we have $(\alpha_2 - \alpha_1) = p_1\alpha_1 + p_0$. Since p_0 will usually have a near-zero value, p_1 will approximately be ratio of the crown width divided by the rosette radius. For two-crown cases, we have $\alpha_2 = p_2\alpha_3 + p_1\alpha_1 + p_0$. As stated before, the dependency on α_3 is low (p_2 is near zero), and like before, p_0 is near zero. Hence, again p_2 is a good indication of the ratio between the first crown and the rosette. Only that in this case it will be indicated as $1 +$ ratio since α_2 is the crown width plus α_1. With this is proven that the optimal proportion is always between 0.65 and 0.75, usually between 0.68 and 0.70. Their similarity points to a validity of the results. It is also observed that the dependency on α_3 is little and negative, meaning that the higher α_3, the smaller α_2. The goodness of fit (whose r^2 is calculated a verification of the goodness of the adjustment) proves both linear and parabolic adjustments appropriate, since it is usually close to 0.99999 and higher. Moreover, for linear adjustments, the 13th inferior value is over 0.999, and for parabolic adjustments, the 16th is over 0.9999. A few cases are shown here, the rest are presented in the appendix. Here, b cases with one crown and two crowns with 4 elements are presented.

3.10.1 Adjustment results for one-crown cases

Since the optimal proportion for a case is not the same as the one for its two-crown counterpart (along its maximal variation and criterion 2, which we will define later) we can affirm that the second crown changes the shape of the curves even ifits precise value α_3 does not affect too much. In conclusion: its presence changes the result, but its precise value does not affect it.

Table 3.1 Adjustment results for one-crown cases.

4E	**type 1 type 0**	**type 2 type 0**	**type 3 type 0**	**type 4 type 0**	**type 5 type 0**
p_1	0.683569908	0.693775343	0.693285007	0.754764074	0.693582651
p_0	-0.014835322	-0.006490713	-0.005586549	-0.02821319	-0.005747557
6E	**type 1 type 0**	**type 2 type 0**	**type 3 type 0**	**type 4 type 0**	**type 5 type 0**
p_1	0.699419006	0.699419006	0.701992941	0.747145291	0.69985529
p_0	-0.00590665	-0.00590665	-0.006206542	-0.022657676	-0.005526055
8E	**type 1 type 0**	**type 2 type 0**	**type 3 type 0**	**type 4 type 0**	**type 5 type 0**
p_1	0.701825963	0.701825963	0.704007907	0.774798469	0.70255688
p_0	-0.005596863	-0.005596863	-0.005826572	-0.032972224	-0.005686963
12E	**type 1 type 0**	**type 2 type 0**	**type 3 type 0**	**type 4 type 0**	**type 5 type 0**
p_1	0.7041073	0.7041073	0.706035086	0.756101823	0.755717653
p_0	-0.005666085	-0.005666085	-0.005878402	-0.024650382	-0.024717687
24E	**type 1 type 0**	**type 2 type 0**	**type 3 type 0**	**type 4 type 0**	**type 5 type 0**
p_1	0.697316181	0.705272161	0.706944907	0.760888331	0.756241747
p_0	-0.002660257	-0.00558706	-0.00576177	-0.026228621	-0.024480946

3.10.2 Adjustment results for two-crown cases

Table 3.2 Adjustment results for two-crown cases.

			1st crown type				
	4E		**1**	**2**	**3**	**4**	**5**
2nd crown type	**1**	p_1	1.7044291	1.7044291	1.7078642	1.6705268	1.6697719
		p_2	-0.0031716	-0.0031716	-0.0030514	-0.0033376	-0.0028602
		p_0	-0.0093358	-0.0093358	-0.0098965	-0.0001454	-0.0009216
	2	p_1	1.7044291	1.7044291	1.7078642	1.6705268	1.6697881
		p_2	-0.0031716	-0.0031716	-0.0030514	-0.0033376	-0.0027438
		p_0	-0.0093358	-0.0093358	-0.0098965	-0.0001454	-0.0009540
	3	p_1	1.7019359	1.7019359	1.7052277	1.6684431	1.6678812
		p_2	-0.0005714	-0.0005714	-0.0005031	-0.0017006	-0.0011116
		p_0	-0.0100269	-0.0100269	-0.0105351	-0.0004649	-0.0012308
	4	p_1	1.6772839	1.6772839	1.6792787	1.6829170	1.6885479
		p_2	-0.0025825	-0.0025825	-0.0025204	-0.0024255	-0.0019403
		p_0	-0.0014264	-0.0014264	-0.0015817	-0.0020208	-0.0043773
	5	p_1	1.7076193	1.7076193	1.6788509	1.6826931	1.6880808
		p_2	0.0002524	0.0002524	-0.0020115	-0.0020918	-0.0013794
		p_0	-0.0121905	-0.0121905	-0.0021855	-0.0027197	-0.0052554

3.11 Scores

Let us define a criterion to find the most advantageous case to use. There are two possible answers:

- The stability criterion: we are searching a configuration in which the effect of the crown(s) improves the result, but doesn't dramatically affect it, since it would result in a fast deviation from the desired result. That is to say, the sensibility of the result must be very slightly dependant on the crown size, thus allowing several different sizes (without much precision in its building) to grant the right solution. Graphically, it can be seen as a very slight negative slope at the cut-off points between the surfaces/curves and the right solution surface/line. This can be spotted at the code as 'criterio_2'. It is the only criterion with a simple physical interpretation. As we can see in the figure:

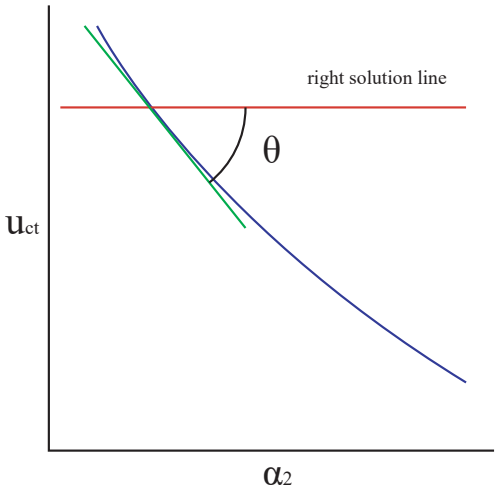


Figure 3.42 Case with fast divergence.

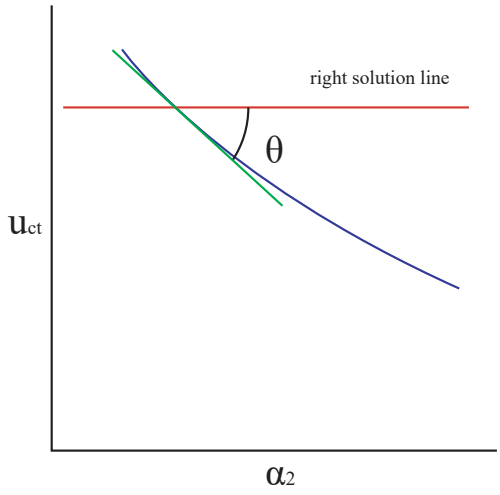


Figure 3.43 Case with slow divergence.

In the figure on the left, the solution diverges rapidly whereas in the figure on the right, the simulation result remains closer to the right solution, hence, it is better to obtain a graphic like that on the right. To model this, we must first calculate the angle of the curve (its slope) in the cut-off points, then average them out, express the result as a value between 0 and 1 (that is the reason why the criterion is expressed as $(\pi/2-\theta)/(\pi/2)$, since the best case would be having a near-zero theta).

To apply the same criterion to two-crown cases we can average the slopes of each cut-off point (in the code as $criterio2$). Yet this would not be a good method. Since the definition domain of the surfaces is triangular (as we can see in the previous figures), there are more cut-off points near the base. These are the cases with small α_1, which allows much more α_2-α_3 configurations to appear, since they have more space to appear. This results in an over-sampling of the region near the base, which tends to have worst results (much higher slopes since they are the first cases to be considered valid, hence not very adequate) and thus artificially worsen the result, since there are more instances of the worst results. To solve this, we can firstly average the results for each α_1 and then average those results, which is more comparable to the method for one crown.

- The convenience criterion: in a more practical sense, we can also achieve good results with another configuration. For example, taking into account how many different α_1 values are able to achieve the right solution. Also, if the regression has a better goodness of fit (for either linear or parabolic adjustments), the right-solution configurations are easier to achieve with enough adjustment precision. Finally, it is understood that using the least possible amount of nodes is preferable. These are in the code: criterio_1, criterio_3a, criterio_3b, criterio_4. The number 1131 is the highest number of nodes used in any case. The convenience criterion is be the average of any of these criteria plus the stability criterion.

  Criterio 1 can be appreciated in the previous images of the typologies: it is evident that the cases with more valid α_1 (the typologies on the right) values are preferable over those that are only valid for few values (the typologies on the left). Criterio 3 a and b do not affect much the final result, since the values are not different among them, but can serve as indicators of the goodness of fit of the adjustment. Since they correspond to the r^2 of the adjustment and are all generally over 0.9999. They are presented in tables in the appendix.

  All the criteria are expressed between 0 and 1 to have an nondimensional value.

We present the results with the improved criterion 2 (which seems more logical). The tables with the normal criterion 2 are analyzed, but can be consulted in the first appendix.

3.12 Table of results for a cases (improved stability criterion)

The tables have a heat-map-like color code affecting all the tables in one page as a whole. Highest-scoring cases are underlined in green, and lowest-scoring cases in red. Orange is for average cases.

Table 3.3 Scores for cases a with improved stability criterion.

		1st Crown				
	4E Stab.	***1***	***2***	***3***	***4***	***5***
2nd Crown	***0***	0.662867074	0.664792432	0.664880699	0.662006348	0.663763655
	1	0.669596687	0.669597507	0.67063037	0.655767673	0.658285083
	2	0.666746609	0.669596687	0.67063037	0.655767672	0.657956754
	3	0.665668339	0.665668339	0.669597475	0.652278565	0.653957958
	4	0.663314223	0.663314223	0.664188574	0.664166621	0.669400217
	5	0.679529819	0.679529819	0.664875825	0.663602237	0.669189583

		1st Type				
	6E Stab.	***1***	***2***	***3***	***4***	***5***
2nd Crown	***0***	0.663900487	0.663900487	0.664808487	0.663437268	0.663609073
	1	0.672372249	0.672372249	0.67336265	0.660677565	0.665432649
	2	0.672372249	0.672372249	0.67336265	0.660677565	0.725558507
	3	0.669517536	0.669517536	0.670529067	0.65737483	0.662026827
	4	0.669042649	0.669042649	0.669803571	0.66833265	0.673140343
	5	0.677229499	0.736839107	0.678472888	0.668293273	0.672097614

		1st Type				
	8E Stab.	***1***	***2***	***3***	***4***	***5***
2nd Crown	***0***	0.662714516	0.662714516	0.663577814	0.66264669	0.663753458
	1	0.669474544	0.669474544	0.670253064	0.664694988	0.669413177
	2	0.669474544	0.669474544	0.670253064	0.664694988	0.669375441
	3	0.666464867	0.666464867	0.667269408	0.661743902	0.666460208
	4	0.668699631	0.719157603	0.669364767	0.669093893	0.669244805
	5	0.668784391	0.668784391	0.669562523	0.668890984	0.668951643

		1st Type				
	12E Stab.	***1***	***2***	***3***	***4***	***5***
2nd Crown	***0***	0.663731231	0.663731231	0.664507975	0.663651941	0.64748072
	1	0.669816482	0.724331233	0.670514614	0.665615869	0.671238851
	2	0.669816482	0.724331233	0.670514614	0.665615869	0.671260098
	3	0.667016146	0.720869028	0.66774757	0.662793084	0.668798475
	4	0.669229598	0.716622919	0.669760242	0.66970683	0.674002723
	5	0.672713457	0.720465668	0.673489099	0.669537201	0.673512846

		1st Type				
	24E Stab.	***1***	***2***	***3***	***4***	***5***
2nd Crown	***0***	0.670563026	0.663725995	0.66448514	0.663651941	0.646842933
	1	0.669758419	0.723107405	0.670399171	0.665200955	0.674874837
	2	0.669758419	0.723107405	0.670399171	0.665200955	0.674874055
	3	0.667202579	0.71989872	0.667883028	0.662478884	0.672819212
	4	0.679086512	0.712351091	0.670190515	0.669600633	0.673898285
	5	0.669966317	0.669966317	0.671234838	0.669743844	0.673977912

3.13 Table of results for a cases (convenience criterion, parabolic adjustment, improved criterion 2)

The linear adjustment is not here presented, since choosing either parabolic or linear adjustment have little final effect: both have similar (and non-decisive) values.

Table 3.4 Scores for cases a with improved convenience criterion.

1st Type

4E Par. Conv.	*1*	*2*	*3*	*4*	*5*
0	0.818455561	0.807222396	0.806359551	0.805642849	0.806523585
1	0.749995655	0.751764206	0.751137676	0.718760524	0.719831905
2	0.751050962	0.751763481	0.751137676	0.718760524	0.719749828
3	0.74989722	0.74989722	0.749995276	0.716119867	0.717423816
4	0.77708693	0.77708693	0.776421344	0.774647057	0.750388102
5	0.69824967	0.69824967	0.749257423	0.775832251	0.750329802

2nd Crown

1st Type

6E Par. Conv.	*1*	*2*	*3*	*4*	*5*
0	0.805672047	0.805672047	0.804573127	0.804232081	0.748719416
1	0.778466929	0.778466929	0.777388666	0.745112673	0.774963593
2	0.778467323	0.778467323	0.777388666	0.745112673	0.73346678
3	0.776427384	0.776426981	0.77535401	0.741634436	0.772343761
4	0.804086217	0.804086217	0.802950174	0.799930094	0.776890977
5	0.779239423	0.794141825	0.778224018	0.802130678	0.776630248

2nd Crown

1st Type

8E Par. Conv.	*1*	*2*	*3*	*4*	*5*
0	0.804051489	0.804051489	0.802499006	0.802266414	0.802541549
1	0.748196191	0.775789999	0.746622417	0.771242931	0.69174163
2	0.748196191	0.775789999	0.746622417	0.771242931	0.71950997
3	0.745675417	0.77326915	0.744108147	0.766967897	0.688792953
4	0.801789972	0.813423243	0.800187936	0.79658371	0.745928874
5	0.747582136	0.747582136	0.746008319	0.79763819	0.774516462

2nd Crown

1st Type

12E Par. Conv.	*1*	*2*	*3*	*4*	*5*
0	0.801653383	0.801653383	0.795216277	0.796328469	0.792284657
1	0.772523324	0.813665967	0.770045337	0.791293084	0.739353998
2	0.772523324	0.813665967	0.770045337	0.791293084	0.73935931
3	0.769170721	0.810147711	0.766701057	0.785282412	0.734765125
4	0.794848491	0.805806167	0.789676182	0.785458029	0.765612339
5	0.772805448	0.812275368	0.742569075	0.788294882	0.773005343

2nd Crown

1st Type

24E Par. Conv.	*1*	*2*	*3*	*4*	*5*
0	0.739788738	0.79369446	0.778853309	0.796328469	0.778421109
1	0.706343145	0.774976441	0.701198291	0.791189356	0.76229367
2	0.706343145	0.774976441	0.701198291	0.791189356	0.762293474
3	0.700399146	0.7688691	0.695264217	0.785203862	0.744538383
4	0.785377242	0.792817333	0.767238057	0.785431479	0.754976168
5	0.733509286	0.733509286	0.728521394	0.788346543	0.762511545

2nd Crown

3.14 Table of results for b cases (stability criterion, improved criterion 2)

Table 3.5 Scores for cases b with improved stability criterion.

		1st Type				
	4E Stab.	***1***	***2***	***3***	***4***	***5***
2nd Crown	***0***	0.662867074	0.664792432	0.664880699	0.661266908	0.665107457
	1	0.669596687	0.669597507	0.67063037	0.655125352	0.657069575
	2	0.666746609	0.669596687	0.67063037	0.655125352	0.656748736
	3	0.665668339	0.665668339	0.669597475	0.651686089	0.652751069
	4	0.663049668	0.663049668	0.663939203	0.662503956	0.668870601
	5	0.680489353	0.680489353	0.664810198	0.663602237	0.669189583

		1st Type				
	6E Stab.	***1***	***2***	***3***	***4***	***5***
2nd Crown	***0***	0.665375549	0.665375549	0.666223756	0.661735864	0.662289038
	1	0.672906982	0.672906982	0.67380076	0.659856742	0.665176447
	2	0.672906982	0.672906982	0.67380076	0.659856742	0.582605066
	3	0.669961078	0.669961078	0.670906241	0.656584929	0.661811744
	4	0.668389201	0.668389201	0.669169402	0.667477718	0.672320368
	5	0.676801156	0.676801156	0.678054659	0.667583281	0.672485301

		1st Type				
	8E Stab.	***1***	***2***	***3***	***4***	***5***
2nd Crown	***0***	0.662004178	0.662004178	0.662831016	0.661149374	0.662213948
	1	0.669062701	0.669062701	0.669844974	0.663456302	0.669295302
	2	0.669062701	0.669062701	0.669844974	0.663456302	0.376978169
	3	0.666102136	0.666102136	0.666905746	0.660550301	0.666358192
	4	0.667103793	0.667103793	0.66781731	0.667919264	0.668005861
	5	0.668656835	0.668656835	0.669470164	0.668125481	0.668525596

		1st Type				
	12E Stab.	***1***	***2***	***3***	***4***	***5***
2nd Crown	***0***	0.661888992	0.661888993	0.662711356	0.661291459	0.66143681
	1	0.669140813	0.669140813	0.669855879	0.664483703	0.668260462
	2	0.669140813	0.669140814	0.669855879	0.664483703	0.667179674
	3	0.66637898	0.66637898	0.667120684	0.661721335	0.665178952
	4	0.667928569	0.667719024	0.668568954	0.668519116	0.66740647
	5	0.672300085	0.672300085	0.669745113	0.668524538	0.668126538

		1st Type				
	24E Stab.	***1***	***2***	***3***	***4***	***5***
2nd Crown	***0***	0.655531694	0.661351013	0.662128576	0.660539344	0.660601902
	1	0.669172468	0.669172468	0.669857606	0.664374929	0.668369872
	2	0.669172468	0.669172468	0.669857606	0.664374929	0.667288907
	3	0.666673062	0.666673062	0.667385278	0.661711122	0.665189219
	4	0.677961468	0.668418069	0.6691373	0.667961014	0.667964108
	5	0.66926204	0.66926204	0.669941234	0.668120592	0.668530488

3.15 Table of results for b cases (convenience criterion, parabolic adjustment, improved criterion 2)

Table 3.6 Scores for cases b with improved convenience criterion.

1st Type

4E Par. Conv.	*1*	*2*	*3*	*4*	*5*
0	0.818455561	0.807222396	0.806359551	0.799047915	0.805534059
1	0.749995655	0.751764206	0.751137676	0.711526582	0.690865815
2	0.751050962	0.751763481	0.751137676	0.711526582	0.690785624
3	0.74989722	0.74989722	0.749995276	0.708898387	0.688459903
4	0.769947429	0.769947429	0.769285637	0.757653135	0.743845449
5	0.669827596	0.669827596	0.747914776	0.775832251	0.750777521

2nd Crown

1st Type

6E Par. Conv.	*1*	*2*	*3*	*4*	*5*
0	0.80515901	0.80515901	0.804044809	0.794523248	0.741537457
1	0.777274735	0.777274735	0.776171496	0.73562364	0.714037902
2	0.777274735	0.777274735	0.776171923	0.73562364	0.688663835
3	0.775212002	0.775212002	0.774122033	0.732153133	0.71142837
4	0.794639055	0.794639055	0.793507832	0.776727871	0.767623209
5	0.744502214	0.744502214	0.743489332	0.787364343	0.776727197

2nd Crown

1st Type

8E Par. Conv.	*1*	*2*	*3*	*4*	*5*
0	0.797463886	0.797463886	0.795902248	0.785313825	0.794863809
1	0.741682986	0.741682986	0.740110148	0.754576071	0.686407116
2	0.741682986	0.748535329	0.740110148	0.754576071	0.638822453
3	0.739174488	0.739174488	0.745343499	0.750312401	0.683462404
4	0.785033894	0.785033891	0.783443948	0.761144179	0.729261931
5	0.740918956	0.740918956	0.739353939	0.780426499	0.740444039

2nd Crown

1st Type

12E Par. Conv.	*1*	*2*	*3*	*4*	*5*
0	0.791909008	0.791909008	0.784599127	0.716973268	0.725626055
1	0.707736078	0.707736078	0.733040101	0.684688282	0.723578466
2	0.707736078	0.707736078	0.733040101	0.684688282	0.7290554
3	0.759948657	0.759948657	0.757481562	0.678692705	0.717945134
4	0.771755888	0.771703117	0.766611006	0.675234604	0.685198179
5	0.757671175	0.757671175	0.754379931	0.705518897	0.708145656

2nd Crown

1st Type

24E Par. Conv.	*1*	*2*	*3*	*4*	*5*
0	0.71288085	0.714335635	0.726392689	0.666608398	0.699555499
1	0.71098593	0.683208152	0.70585217	0.631831845	0.705922379
2	0.71098593	0.683208152	0.70585217	0.631831845	0.705652138
3	0.70505604	0.70505604	0.69992905	0.620555843	0.715663613
4	0.729172034	0.699008335	0.683268117	0.611434532	0.6678015
5	0.707029177	0.707029177	0.701893951	0.647283447	0.71016237

2nd Crown

3.16 Analysis of results, improved criterion 2

Before analysing whether there is a preferable option, a caveat about a strange phenomenon should be made. Cases a with 12 elements involving type 2 and type 4 configurations, and cases a 24 elements (specially those involving type 2 and type 4 configurations) score higher than expected in stability. As seen in the appendix, those crowns have an evident lack of beneficial traits. They were not expected to score high beforehand since, after a visual inspection, none could be considered valid due to the abrupt transition inside the half hexagon. This probably happens not because of the positive traits of the configuration, but because of two causes. Firstly, some artifacts appear in the graphics, invalid cases with near zero slope, which affects the result. And secondly, because the parameters add little variation to the final result and hence it is not really dependent on crown element type or proportion. That is to say, the error due to the invalid shape cannot be corrected by a proper crown proportion, and only coincidentally meets the correct value. The fact that they are the only two-crown configurations whose criterion 1 scores 1 endorses this conjecture, since it indicates that the result is not really dependent on the α_1 value. In consequence, these configurations will be dismissed. Other caveats would include that not all criteria are equally determining, since criteria 3 (linear and parabolic) only slightly affect the result since the difference between the highest and lowest value is in the decimals, while criteria 1 and 2 are decisive. Also, using the goodness of fit can be misleading: fewer valid cases will be able to score higher, since they are more likely to align well, thus not rendering a trustworthy result (even adding criterion 1, which takes into account the valid cases). It has been used only to compare the result to the stability criterion. All in all, criteria are bound to analyze what is happening and should only be used as decision tools together, not as clear demonstrations on their own.

Now we can analyze results case by case. Criteria were not intended to compare a cases and b cases, since the basis of the study relies on the similarity of the region inside the half-hexagon. In any case, the difference between respective cases is usually negligible.

For a cases with stability criterion, we can assert that best cases involve either a type 5 first or second crown, or a type 4 second crown. In contrast, type 4 and 5 first crowns do not usually improve the result independently from the type of the second crown, and can even make it worse than a one-crown case. Type 1, 2 (and to a least extent 3) second crown seem to improve the result of its respective one-crown cases, specially for few elements. Type 4 second crowns tend to worsen the result, while type 5 second crowns improve it to the biggest extent with few elements (under 8). Independently, one-crown cases seem to reach a peak for 8 elements and types 2 and 3. Plotting the score versus the number of singular elements indicates that maximum, minimum, and average values stagnate after 12 elements, and in general low values rise and high values descend.

As for a cases with convenience criterion, it is one-crown cases which score higher, due slightly to criterion 4, which penalises cases with more nodes, and mainly to criterion 1, as we will see. Criterion 1 has a small bias toward one-crown configurations, which indicates that only one crown allows more liberty to reach, at some point, the desired result. Firstly, because they need space to exists, and thus impose a top limit to α_1, and secondly, because the dual dominance zone may be better partially meshed by a non-linear element (non-singular linear stress representation zone, for instance the linear behaviour of one side of our singular element) than completely by linear elements. This may explain why type 4 second crowns improve the result whereas type 4 first crowns (which places its elements much closer to the crack tip) worsen it: they allow more proximity of crowns (that is, for the singular elements to take more space) and mesh with more nodes the transition part, but still do not improve one-crown cases. Among the best cases, 4 elements and type 1 or 2 cases seem to be the best option. Plotting the evolution of the score versus the number of singular elements reveals no clear tendency here, although removing maximum values indicates a certain preference towards 6 element cases.

For b cases with stability criterion, tendencies are much clearer. Type 4 second crowns tend to worsen or under-improve the result of other second crowns, and even one-crown cases for few elements. Type 1, 3 and specially 2 second crowns considerably improve the result for few elements. Type 5 second crown, specially for few elements, considerably improve the result. Yet the improvements are of little relevance when the number of singular elements increases, that is to say, the lower values increase and the higher values decrease, scoring closer to the average, which does not change much. However, with less distortion, one is more likely to find a good configuration for few elements. Even one-crown cases score less as the number of singular elements grow. This may explain a lack of fast-pace convergence for this configurations due to poor mesh quality. Plotting the results as before reveals that max and minimum values diverge considerably, although the average does not improve much after 8 elements.

Finally, b cases with convenience criterion score in a similar fashion to a cases with convenience criterion, only much more affected by the augment of the number of nodes. Single-crown cases are preferable, type 1/2/3 type 1/2/3 score lower, type 4 second crown worsen the result, and the only remarkable change is a good effect of the type 3 second crowns. In later cases, it probably scored below type 2 because elongates the shape of elements closer to the side that approximates singularity, instead of narrowly approaching the singularity in the singularity-affected zone. The unexpectedly high score of 24 Elements type 1 type 4 may be due to the mesh refinement overtaking criterion 4 penalisation on the number of nodes. Plotting the results, as expected, worsens with the number of nodes, but in average, 6 elements improves over 4 elements.
As a final phenomenon, it is worth noting that although the second crown improves the result, its precise size does not cause significant changes in the result. It is only necessary for it to be present. The slightest change in α_1 and α_2 causes major changes in optimal α_3 value, and the inverse is true: α_3 value only affects the solution by a little bit. Maximum changes due to α_3 in the final result for a fixed α_2 are presented in a table in the appendix.

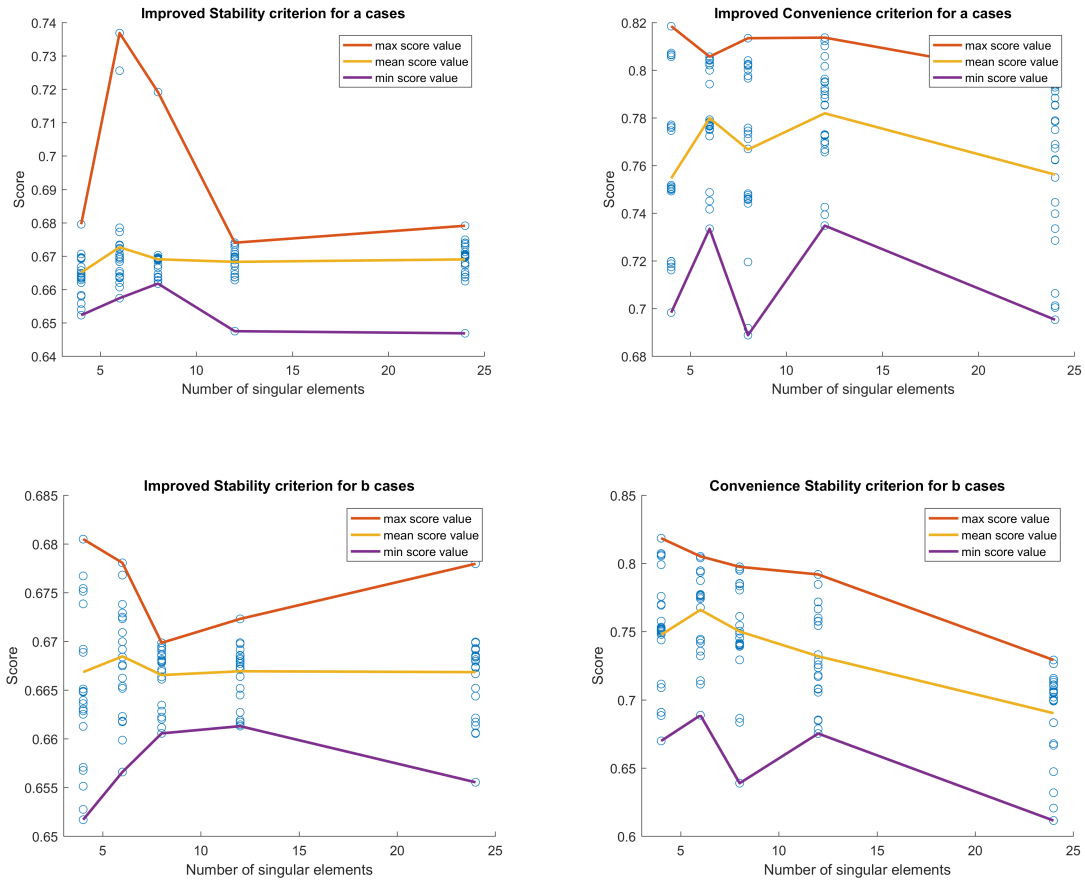


Figure 3.44 Evolution of results with improved criterion 2.

By this graphs, we can conclude that the element is very sensitive to angle warping.

3.17 Analysis of results, criterion 2

Results here are different, since a second crown does not improve the result for few singular elements, due to the over-sampling in the poorest regions. Overall, the same principles apply, although it is worth noting how type 4 second crowns are less penalised for higher number of elements in the convenience criterion, in contrast to the big penalisation in the previous section. Evidently, the main difference is in the quality of the two-crown cases for few elements, which is much lower here.
The evolution graphics are displayed to note that the depression due to eight-element cases is not longer present if the quality of the mesh is measured this way. Also, some of the highest-scorers for 12 and 24

elements are not present here, which is logical, since the average (and most cases) carried on much the same way.

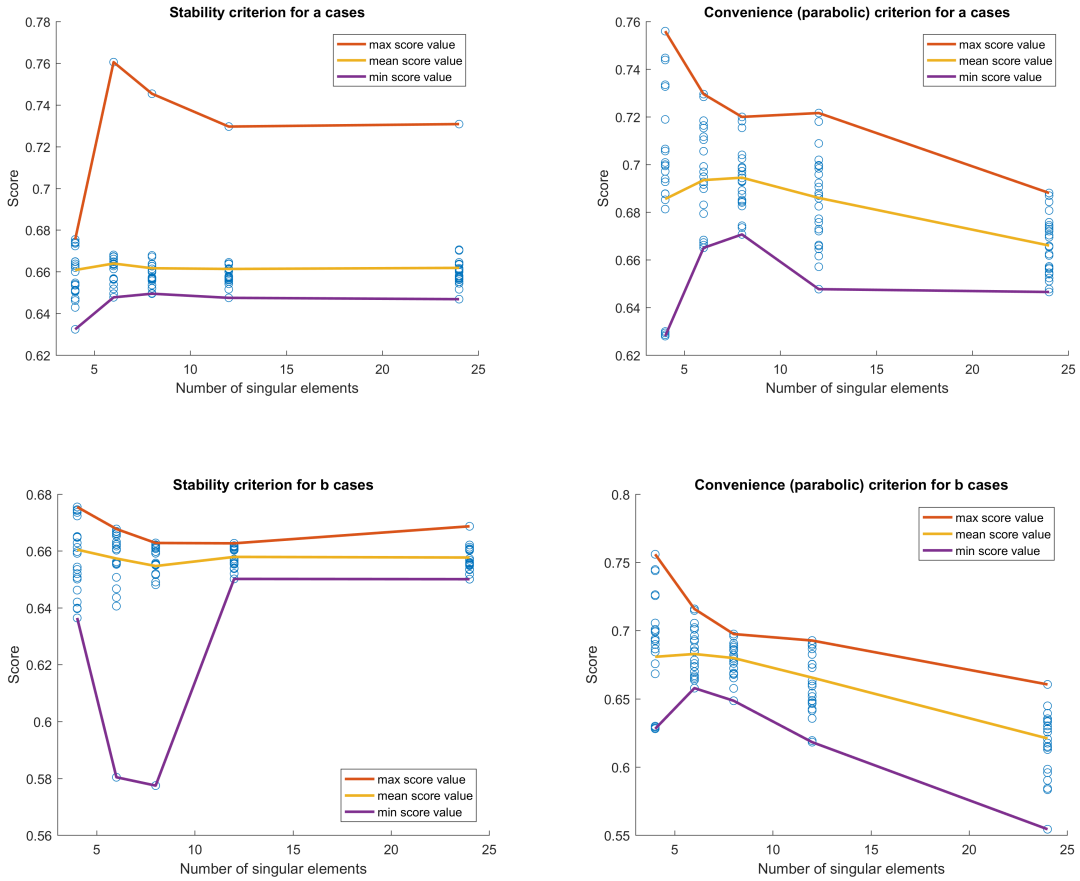


Figure 3.45 Evolution of results with normal criterion 2.

3.18 Final analysis

With this in mind, we will choose to continue studying the highest-ranking configuration for each a/b case and stability/convenience improved criteria: 6E type 2 type 5 a, 4E type 1 type 0 a, 4E type 1 type 5 b, and 4E type 1 type 0 b. Since 4E type 1 type 0 a and 4E type 1 type 0 b are the same case, we will be omitting 4E type 1 type 0 b. To get a broader look, we will also analyze the following cases: 6E type 5 type 2 a (second best a stability case), 8E type 2 type 4 a (second best a convenience case, whose validity is not clear), 6 E type 3 type 5 b (third best b stability case, since the second is presumably similar to the first and scores low in criterion 1), and 6 E type 2 type 0 b (the best differentiated b convenience case).
This does not mean that these configurations will give better results, but that they will achieve them more easily (with less nodes, or without too much precision needed for α_1). As it has been shown, every configuration can achieve the right solution, albeit with a different adjustment. Hence, we are taking those configurations only as a representative selection which would be easy to study since results are very precision-sensitive. In the summary, there is a comparison of each case's linear adjustment, to check if optimal proportions are similar and thus there is a preferable one.
Once the effect of the crowns is identified, we can make a few changes in the meshing options. Until now, the region inside the half hexagon had been meshed freely, in order to avoid the automatic appearance of crowns that happens with a structured meshing. Since we wanted to find out the effect of the designed crown, no other could have been taken into account (specially one that could sporadically appear thanks to the algorithm). Furthermore, all our configurations have been proven advantageous when employed with a similar amount of meshing seeds around (dependent only on the crown). Since we will be comparing similar instances and

focusing on appropriate surroundings, we can adjust the number of hexagon seeds to the space between the last-crown circle and the half-hexagon. The following chapter explains those minor differences in the code. The only remaining problem is how to create a case the optimal proportion. As we have seen, the maximum change in the crack tip displacement due to the second crown is not very dependant on α_3. However, there is technically a dependency which, since α_2 values that are very close between themselves have very different α_3 values which result in the right u_{ct}, may make optimal α_3 have a big variation depending on α_1 and α_2. From now on, we will be using the proportion between crowns that results in the right crack tip displacement. Hence, we only have two independent parameters: two among α_i values. The third one will be determined by these two values. As explained before, it is much easier to extract α_2 from the other α values than doing so with α_3. Hence, α_1 will be the parameter, α_3 will depend on α_1 (the different decisions that can be made will be discussed), and α_2 will be extracted via the previously calculated adjustment planes.
As for the following graphs, stability criterion for a cases reaches its peak for 6 elements, but removing the highest values would place it on 4 elements. Convenience criterion for a cases has its peak for 8 elements. For b cases the peak is on 6 elements, but similarly, removing the highest scoring value would make the remaining six-element cases not so higher or even lower than the four-element cases. In any case it is more or less consistent that a higher number of elements reduces (or does not significantly augment) either criteria. Mean values roughly follow the higher-scoring tendency.

3.19 Summary: best options

Table 3.7 Cases we will study.

Reason for the choice	Chosen config.	note	free	structured
Case a improved stability criterion #1	6E type 2 type 5 a,		x	
Case a improved convenience criterion #1	4E type 1 type 0 a	a and b coincide	x	
Case b improved stability criterion #1	4E type 1 type 5 b	low criterion 1 score	x	x
Case b improved convenience criterion #1	4E type 1 type 0 b	a and b coincide	x	
Case a improved stability criterion #2	6E type 5 type 2 a		x	x
Case a improved convenience criterion #2	8E type 2 type 4 a		x	x
Case b improved stability criterion #3	6 E type 3 type 5 b	#2 is similar to 4 E type 1 type 5	x	x
Differentiated case b improved convenience criterion #1	6E type 2 type 0 b	For all of the previous, a and b coincide	x	

As seen in the next page, when restrictions are relaxed in order to get a structured region inside the half-hexagon, we automatically obtain one or more non-circular crowns around the last one imposed by our cuts. Besides, the case with 24 E does not seem to be appropriate due to its shape, although the sheer amount of elements may compensate for it. However, we are keeping it in order to get a sample of its behaviour, excluding the structured case since its shape is even more warped. Finally, we can now present the image of the crack tip surroundings corresponding to each chosen configuration.

Table 3.8 Influence of the second crown, expressed in units of distance.

Cases b	4E	6E	8E	12E	24E
type_1_type_1	1.3778e-04	1.1123e-04	1.0603e-04	9.0118e-05	9.6040e-05
type_1_type_2	1.3778e-04	1.1123e-04	1.0603e-04	9.0118e-05	9.6040e-05
type_1_type_3	2.7912e-04	1.2520e-04	1.3838e-04	1.2088e-04	1.1101e-04
type_1_type_4	1.1887e-04	1.3014e-04	1.4234e-04	1.5679e-04	1.5227e-04
type_1_type_5	1.0917e-04	9.5634e-05	9.3167e-05	9.1049e-05	9.9930e-05
type_2_type_1	1.3778e-04	1.1123e-04	1.0603e-04	9.0118e-05	9.6040e-05
type_2_type_2	1.3778e-04	1.1123e-04	1.0603e-04	9.0118e-05	9.6040e-05
type_2_type_3	2.7912e-04	1.2520e-04	1.3838e-04	1.2088e-04	1.1101e-04
type_2_type_4	1.1887e-04	1.3014e-04	1.4234e-04	1.4681e-04	1.5037e-04
type_2_type_5	1.0917e-04	9.5634e-05	9.3167e-05	9.1049e-05	9.9930e-05
type_3_type_1	1.3800e-04	1.0413e-04	1.0344e-04	8.9275e-05	9.4159e-05
type_3_type_2	1.3800e-04	1.0413e-04	1.0344e-04	8.9275e-05	9.4159e-05
type_3_type_3	2.8268e-04	1.2811e-04	1.4135e-04	1.2455e-04	1.1429e-04
type_3_type_4	1.1770e-04	1.2755e-04	1.4257e-04	1.5501e-04	1.5060e-04
type_3_type_5	2.4078e-04	9.5346e-05	9.2246e-05	8.9324e-05	9.7086e-05
type_4_type_1	2.2717e-04	1.8352e-04	1.7795e-04	1.6042e-04	1.6743e-04
type_4_type_2	2.2717e-04	1.8352e-04	1.7795e-04	1.6042e-04	1.6743e-04
type_4_type_3	2.1585e-04	1.7504e-04	1.8583e-04	1.4262e-04	1.4335e-04
type_4_type_4	1.5373e-04	1.4415e-04	1.4709e-04	1.4843e-04	1.5260e-04
type_4_type_5	1.5474e-04	1.4495e-04	1.5061e-04	1.4163e-04	1.4978e-04
type_5_type_1	2.4010e-04	1.8403e-04	1.2155e-04	1.2583e-04	7.6300e-05
type_5_type_2	2.3040e-04	9.1455e-03	5.2476e-02	1.0682e-04	7.6260e-05
type_5_type_3	2.2653e-04	1.6306e-04	1.3808e-04	7.4671e-05	3.2724e-05
type_5_type_4	1.1885e-04	9.7706e-05	1.0872e-04	1.2701e-04	9.0237e-05
type_5_type_5	1.1890e-04	1.0698e-04	9.0895e-05	6.6740e-05	7.3464e-05

3.20 Max change due to the second crown

As we can see in the following table, the average of the maximum change in the result produced by the second crown (measured over the level curve with $\alpha_2 = cte$ and as closest possible to the optimal value) usually affects the fourth decimal by one or two decimals. In contrast, the effect of the second crown is much bigger: it is on the units (can lower the result from the correct solution more than $1ul$).
The table is presented for the cases b, but cases a have the same behavior. It is also observed how as the number of elements grow, the effect of the third crown is also reduced.

3.21 Chosen configuration graphic results

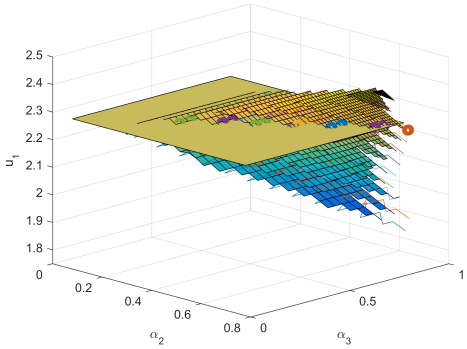


Figure 3.46 Increasing α_1 surfaces for 6E type 2 type 5 a.

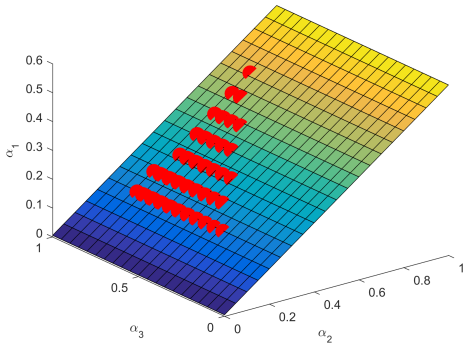


Figure 3.47 adjustment plane for 6E type 2 type 5 a.

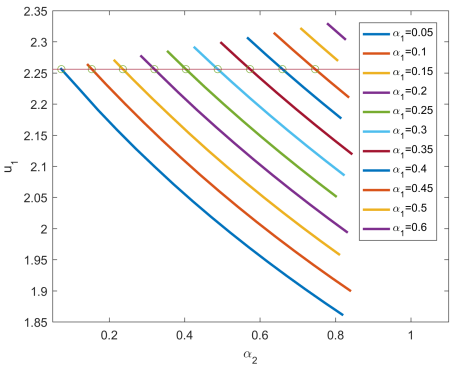


Figure 3.48 Increasing α_1 surfaces for 4E type 1 type 0 a.

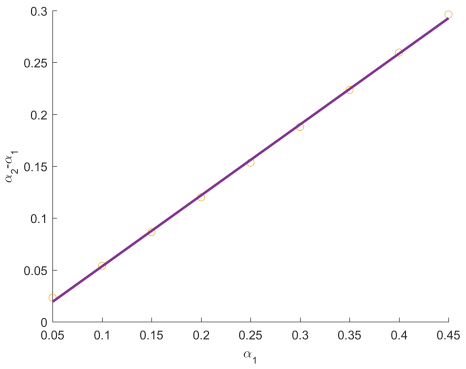


Figure 3.49 adjustment line for 4E type 1 type 0 a.

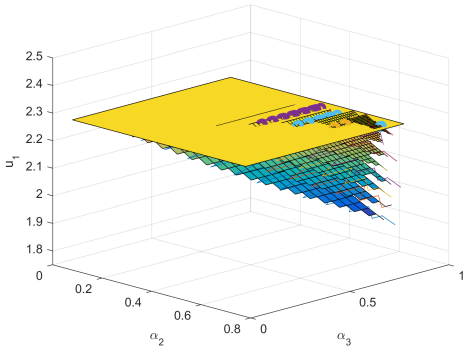


Figure 3.50 Increasing α_1 curves for 4E type 1 type 5 b (there is a considerably smaller number of points, makes the tangent in cut-off points flatter in average).

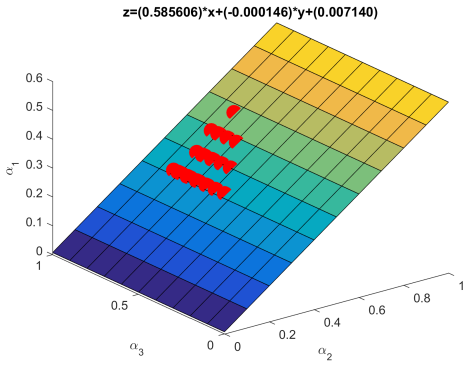


Figure 3.51 adjustment plane for 4E type 1 type 5 b.

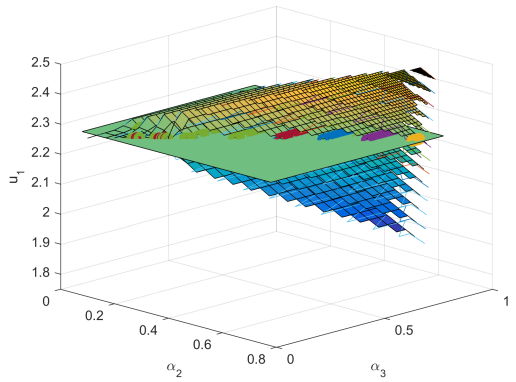


Figure 3.52 Increasing α_1 surfaces for 6E type 5 type 2 a.

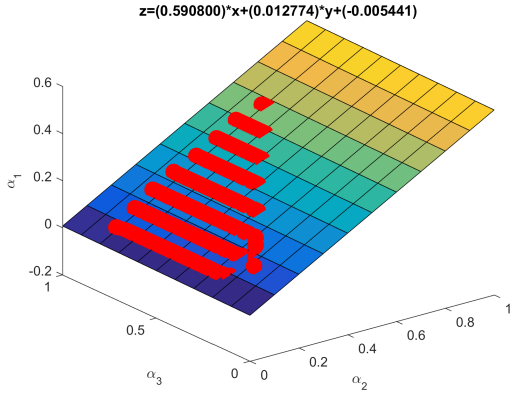


Figure 3.53 adjustment plane for 6E type 5 type 2 a.

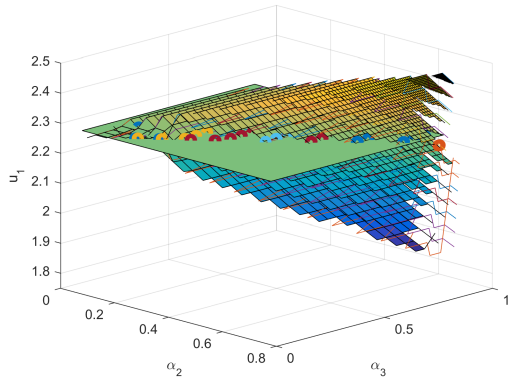


Figure 3.54 Increasing α_1 surfaces for 8E type 2 type 4 a.

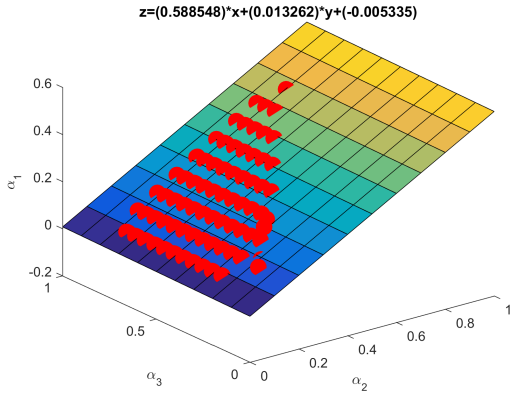


Figure 3.55 adjustment plane for 8E type 2 type 4 a.

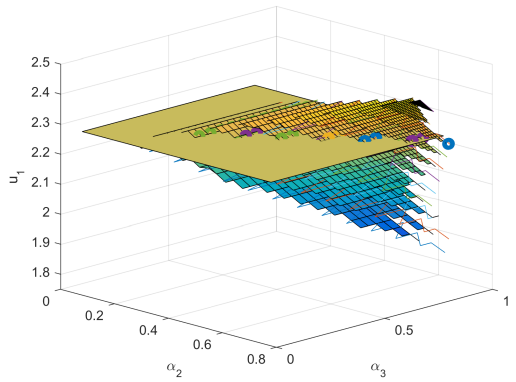


Figure 3.56 Increasing α_1 curves for 6E type 3 type 5 b.

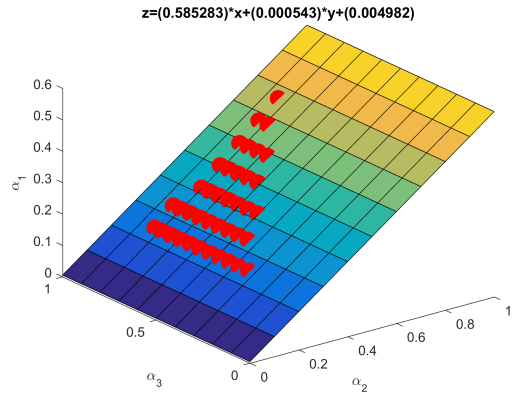


Figure 3.57 adjustment plane for 6E type 3 type 5 b.

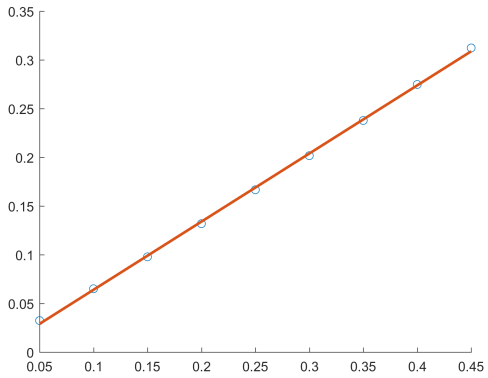


Figure 3.58 Increasing α_1 surfaces for 24E type 1 type 4 b, where some invalid cases warp low α_1 surfaces for low α_2 values.

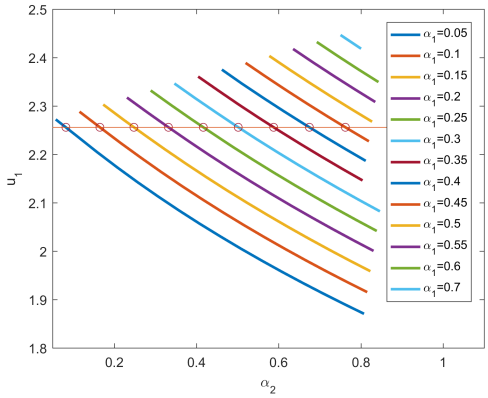


Figure 3.59 adjustment plane for 24E type 1 type 4 b.

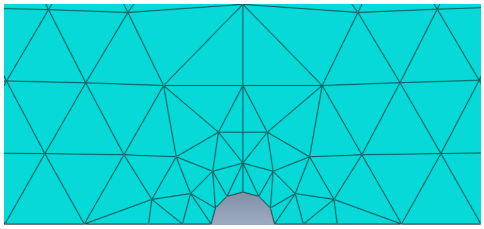


Figure 3.60 Chosen configuration: 6E type 2 type 5 a.

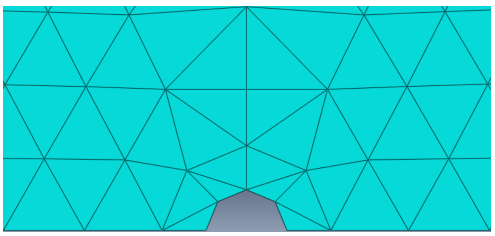


Figure 3.61 Chosen configuration: 4E type 1 type 0 a.

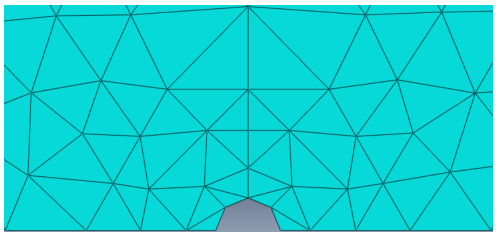


Figure 3.62 Chosen configuration: 4E type 1 type 5 free meshing in half-hexagon area.

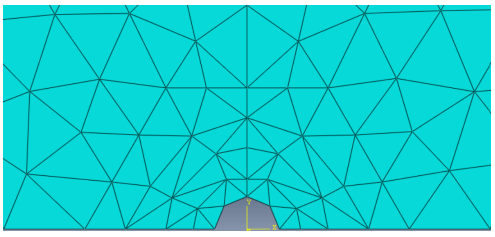


Figure 3.63 Chosen configuration: 4E type 1 type 5 structured meshing in half-hexagon area.

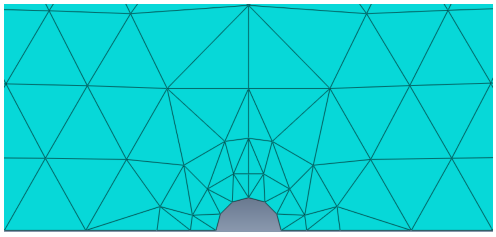


Figure 3.64 Chosen configuration: 6E type 5 type 2 a with free meshing.

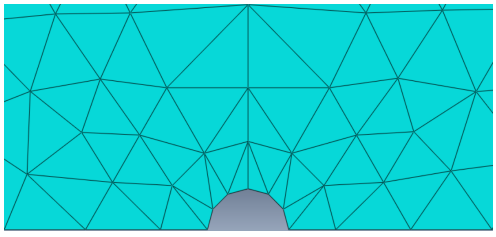


Figure 3.65 Chosen configuration: 6E type 2 type 0 b with free meshing.

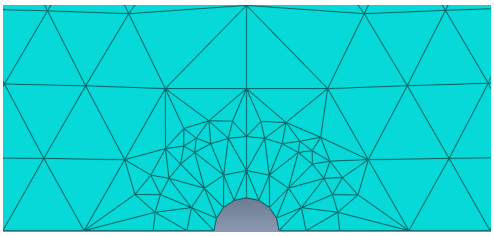


Figure 3.66 Chosen configuration: 8E type 2 type 4 a with free meshing.

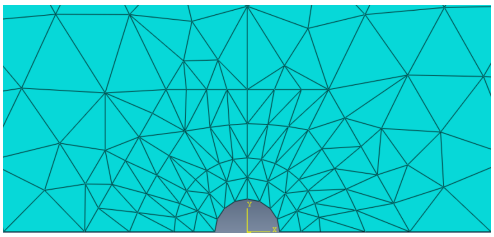


Figure 3.67 Chosen configuration: 8E type 2 type 4 a with structured meshing.

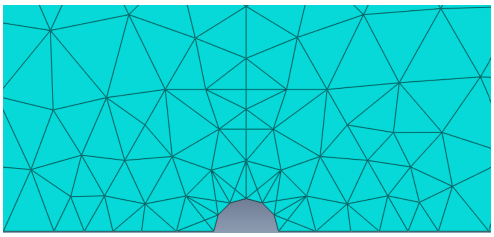


Figure 3.68 Chosen configuration: 6E type 3 type 5 b with free meshing.

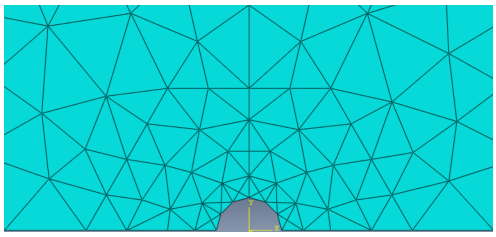


Figure 3.69 Chosen configuration: 6E type 3 type 5 b with structured meshing.

4 Second Parametric Study: optimal α_1 vs hexagon side-length proportion

Now, with the optimal proportion among α_i values, we should find out if there is an optimal value or range of values for α_1 to calculate the right solution. Even when theoretically every configuration is close enough, there may be minor differences in precision. Hence, there are two possible studies:

Element-Hexagon proportion keeping the hexagon side-length constant, varying α_1 values. The effect of the hexagon side-length also should be taken into account (the same study should be made for several values).

Hexagon-crack proportion varying the hexagon side-length in proportion to the crack. The effect of the seeds should also be taken into account (the same study should be made for several configurations).

4.1 Element-Hexagon proportion

Now, as explained before, the only parameter is α_1. For one-crown cases, α_2 depends directly on α_1. And for two-crown cases, α_3 is $m\alpha_1$ (with the effect of m being studied), and α_2 chosen according to the adjustment plane.
The code is basically the same as before (it is added in the appendix for closer examination), but with a minor difference: all the cuts are done in the same step (to fix the numeration problem). Besides, only b cases will be slightly modified in order to achieve an even better meshing. Using a variable amount of elements between the last crown and the half-hexagon in order to make it always be meshed with elements of roughly the same size as those in the lateral and upper sides of the hexagon can be achieved by a simple formula: using $(1-\alpha_3)$ times the number of seeds in the oblique sides of the hexagon. Upper segments will be seeded in a similar fashion. This, however, as will be seen later, is the equivalent of changing the behaviour of the mesh for some cases, and it is easier to compute them separately, and is an indicator of whether the result is a consequence of the mesh or is error-dominated. The same cannot be applied to a cases since the half-hexagon regions with only one seed are poorly fit for a structured meshing.
It is worth noting that some cases plot of u_{ct} vs α_{1opt} will exhibit an uneven curve with severe sudden steps. The curve reaches the exact value, but not in a steady, stable, and easily-predictable manner. In those cases, two solutions have been studied. The first one is imposing a structured meshing, which sometimes results in a flattening of the curve, but often stops the curve before reaching the correct value and reduces the range of validity of the result. The second option is using an inverse method to create the crown: using α_2 as a parameter, choosing $\alpha_3 = m\alpha_2$, and α_1 thanks to the adjustment. This has been proven to augment the range of validity of our result, flatten the curve, but not completely solve the problem since sometimes it is not enough to improve the whole curve. Examples of these methods are added in the following sections. It is probably due to numerical motives involving the adjustment.
To keep track of the mesh-qualities, we will calculate for each case: the α_1 values that result in the right solution, the maximum and minimum value of the curve, the value of m, and relevant notes. These will be analysed later on, after graphics are presented.
The range of α_1 in the figures is limited not only by the hexagon region, but also by the acceptable values in the adjustment.

4.1.1 6E type 2 type 5 a

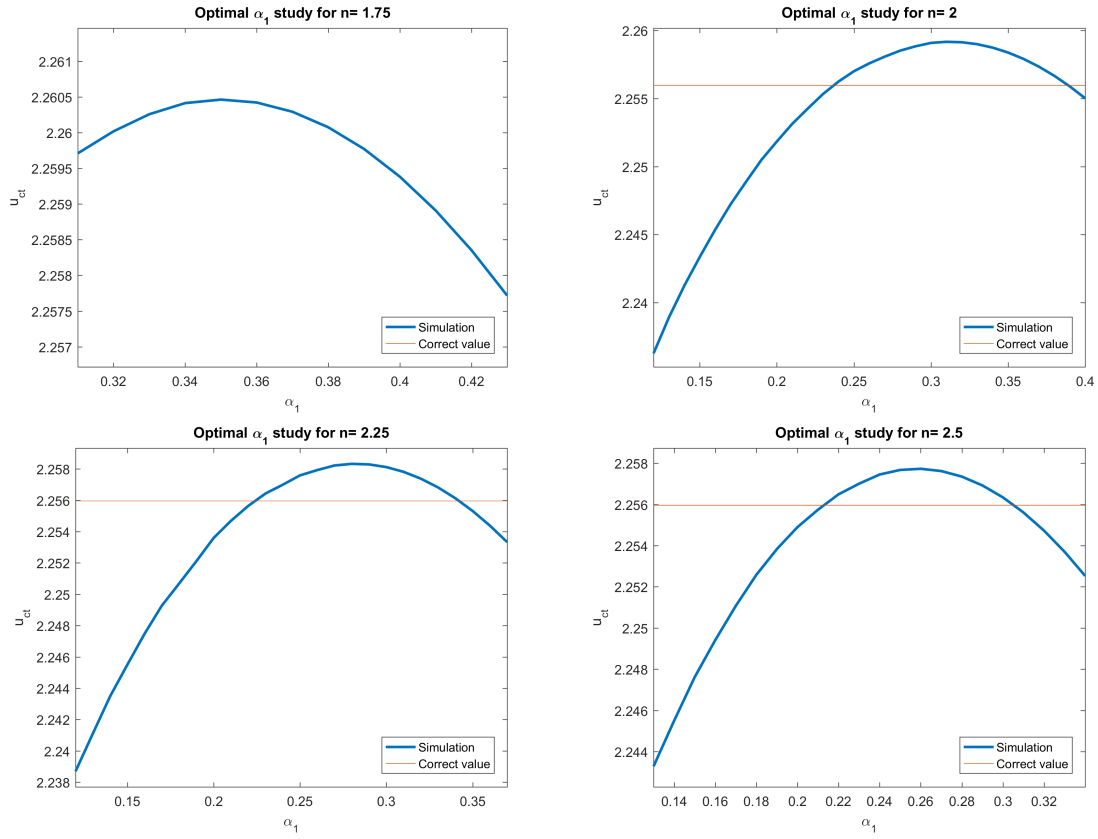


Figure 4.1 6E type 2 type 5 a (Second Study).

As seen in the graphics, the parabola becomes narrower as m increases. Its maximum value descends and its minimum value ascends (it becomes slightly flatter) but not in a relevant manner. The narrowness increase is imposed by the second cut-off point, which moves more than the first. It is also relevant that small m values highly affect the result (in the first image, the correct value is not met), while for higher values it has less impact.

4.1.2 4E type 1 type 0 a

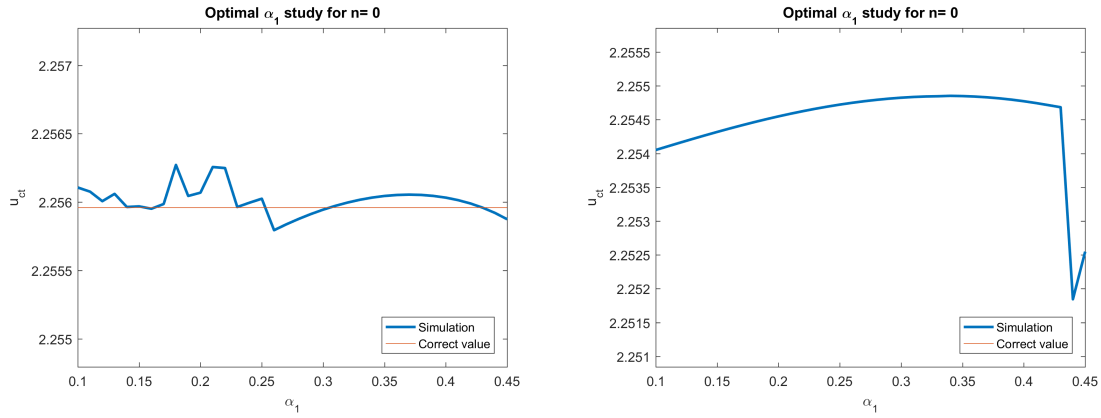


Figure 4.2 4E type 1 type 0 a free and structured (Second Study) respectively.

For this case, there is an irregular zone which does not avoid the curve to meet the desired value, but reduces the range of α_1. A structured mesh does not solve the problem, since the desired values is not met.

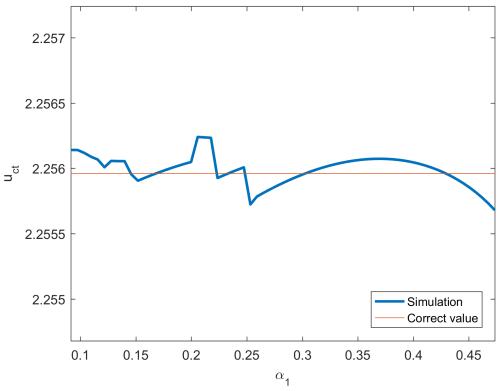


Figure 4.3 4E type 1 type 0 a, alternative (Second Study).

In this case, unlike previous tests for other configurations, the problem is not completely solved: the validity region extends considerably further than before (previously from 0.25 on, now from 0.18 on), but is not fixed. It does not change the flatness of the curve, which is an advantage over the previous two-crown case, since a flatter curve means that it always remains close to the exact solution.
The max value is the one in the regularised section, not counting the irregular section prior. Yet a few validity sections seem to appear roughly between 0.15 and 0.2, where it again matches the desired solution.

4.1.3 4E type 1 type 5 b

We will find analogous results. For the free case, values bellow $m = 2.5$ were invalid. This perhaps has to do with the fact that there were few valid α_1 values for this configuration. Hence, it reaches the value with a good enough flatness but only for certain conditions.Structured meshing fixes the oscillations but makes the result quickly diverge when α_1 grows, as seen in the images. The flatness of the curve is not an optical illusion due to the axis values, as seen in the close-up. And the alternative method, again, makes increase the validity region but does not regularise the whole curve.

free

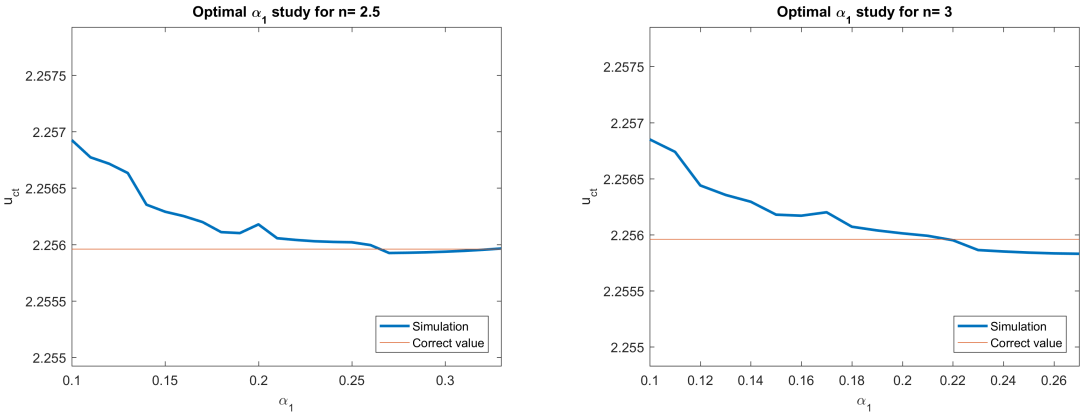


Figure 4.4 4E type 1 type 5 b free (Second Study).

structured

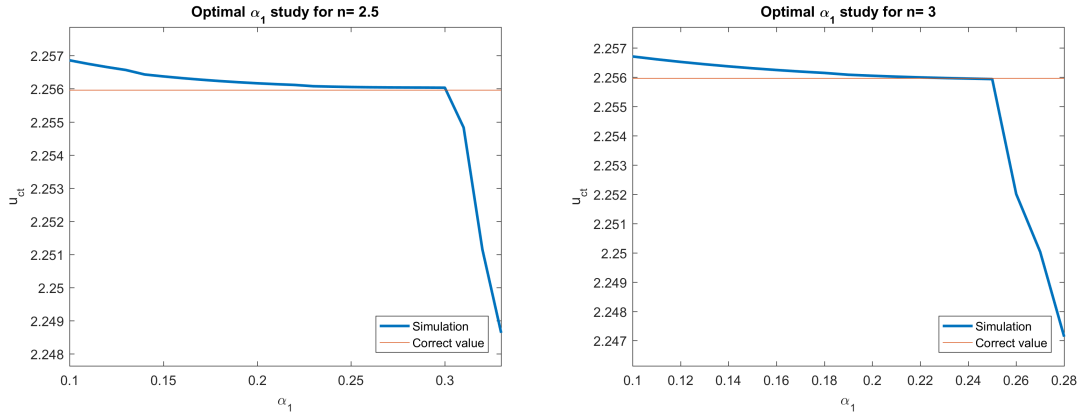


Figure 4.5 4E type 1 type 5 b structured (Second Study).

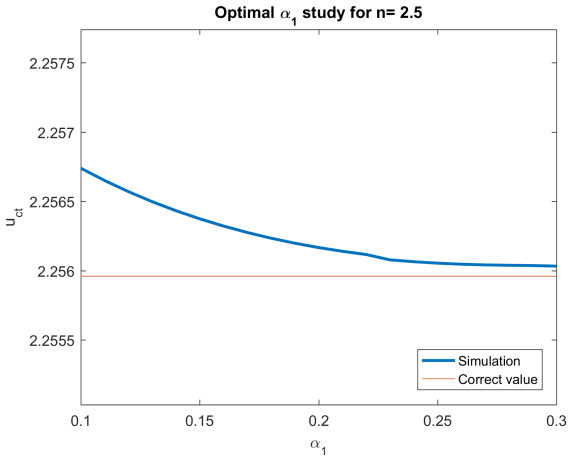


Figure 4.6 Close-up of the curve..

alternative

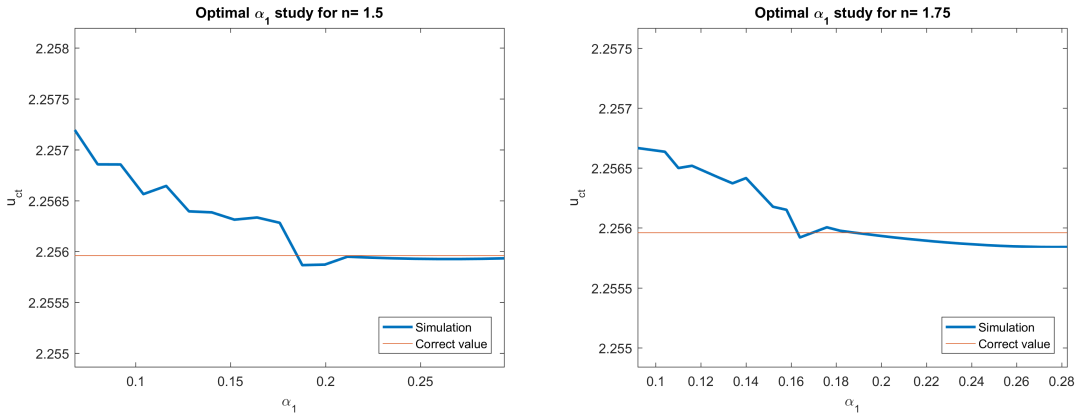


Figure 4.7 4E type 1 type 5 b alternative (Second Study).

4.1.4 6E type 5 type 2 a

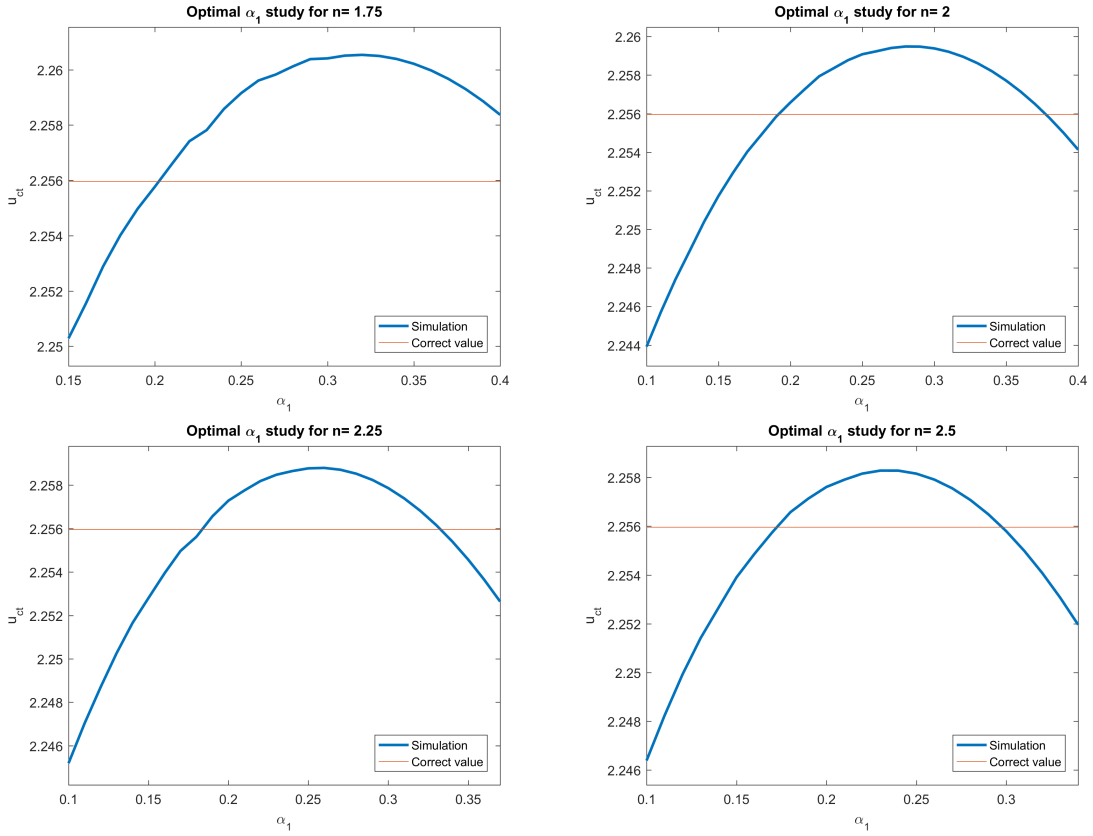


Figure 4.8 6E type 5 type 2 (Second Study).

The conclusions are similar to the first case presented. Here, a structured version of the mesh has also been studied, with virtually no differences. If anything, it appears to sometimes reduce the oscillations, but they are already not very pronounced.

4.1.5 8E type 2 type 4 a

With this last analysis, we can observe that a cases have been less flat than b cases, but also more smooth and stable versus m. Besides, both options: free meshing, with few in-hexagon elements, and structured, with substantially more, have similar results. Hence this case, whose validity was doubtful (since the transition was not completely appropriate), can be considered valid. Only two cases for each option are here shown for comparison.

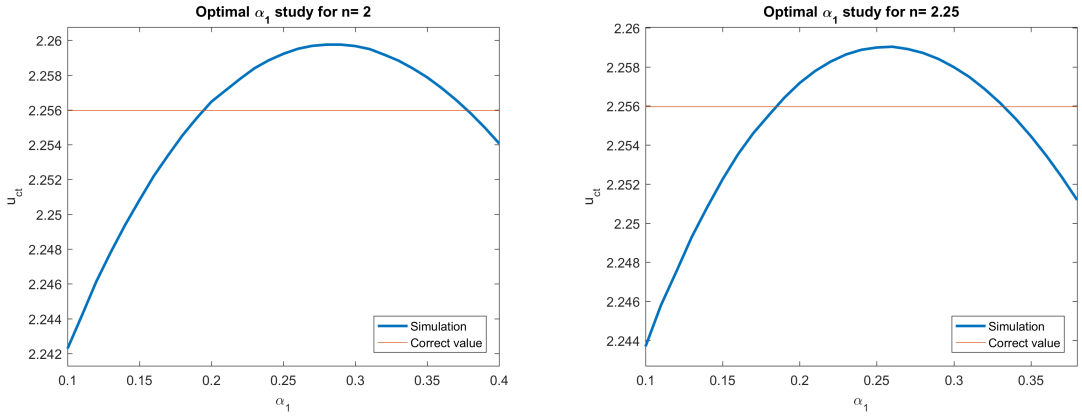


Figure 4.9 8E type 2 type 4 free (Second Study).

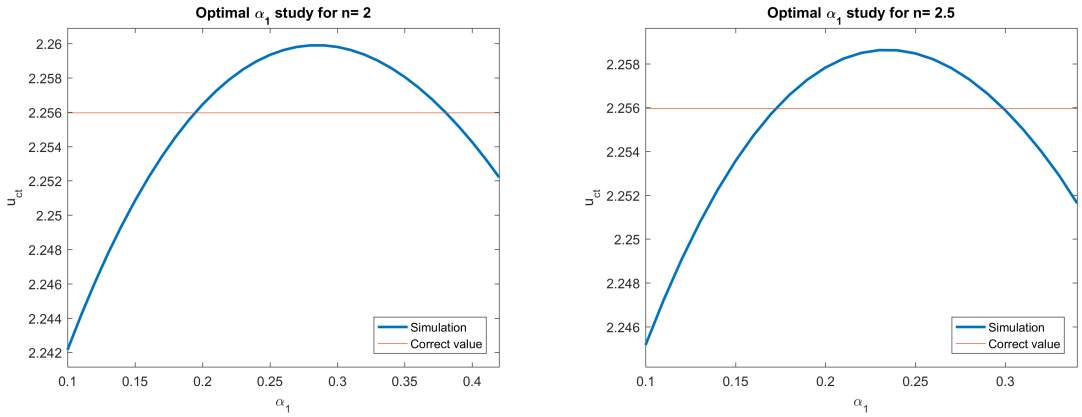


Figure 4.10 8E type 2 type 4 structured (Second Study).

4.1.6 6E type 3 type 5 b

These cases have another beneficial quality: they are the flattest curves we have found, and even intersect the right result at several points. However, the attempts to fix the oscillations result in a major improvement of the result, being one of the closest to the right solution, specially the structured version. Using the mesh options from the previous section does not improve the results either, so the problem does not have to do with the variable number of elements for the half hexagon based on the space between the last crown and the hexagon. In the appendix some graphics are presented using that method, and no improvement is found.

alternative method

In comparison, the curve has lost its best trait (the proximity to the exact solution) but has won manageability, since the exact cut-off points are clear. The last image is the result of applying the alternative method to a structured configuration. There does not seem to be a big difference or improvement: both structured, and alternative plus structured have similar results. As exposed here and later, its effect at best is to flatten the curve while causing convergence issues, and at worst is to lump the curve but clarify the cut-off points.

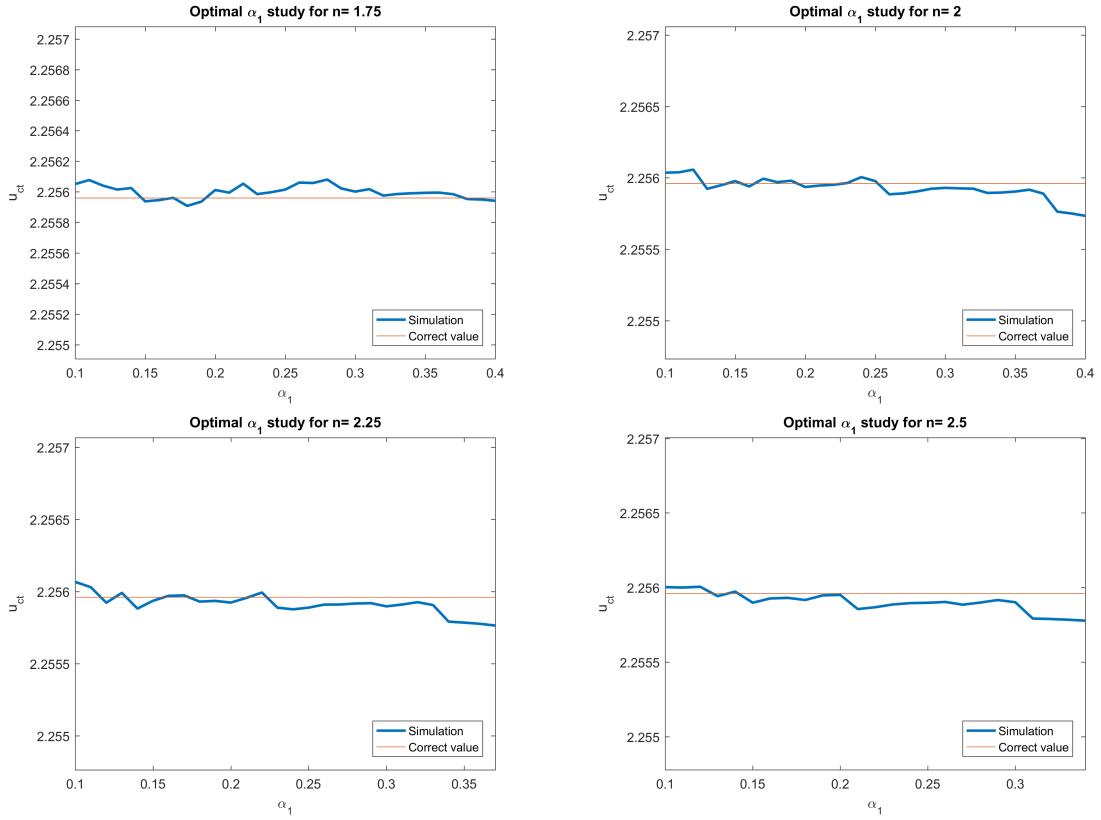


Figure 4.11 6E type 3 type 5 free (Second Study).

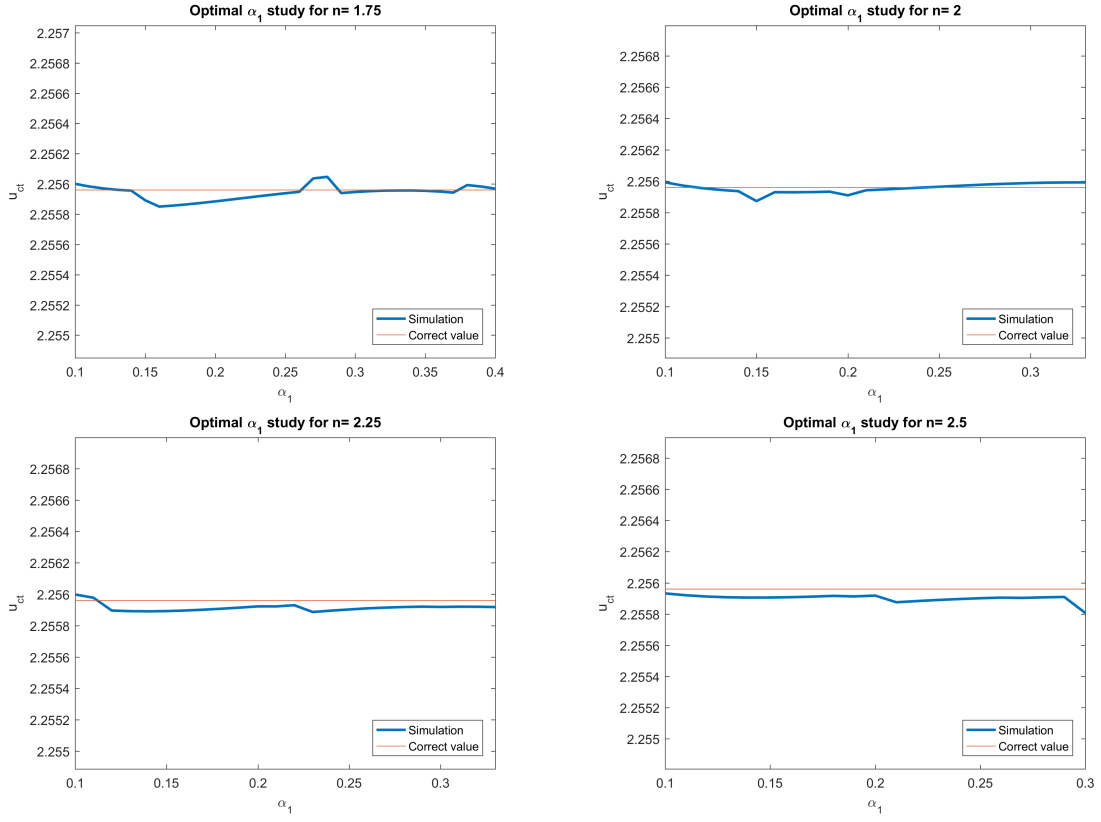


Figure 4.12 6E type 3 type 5 structured (Second Study).

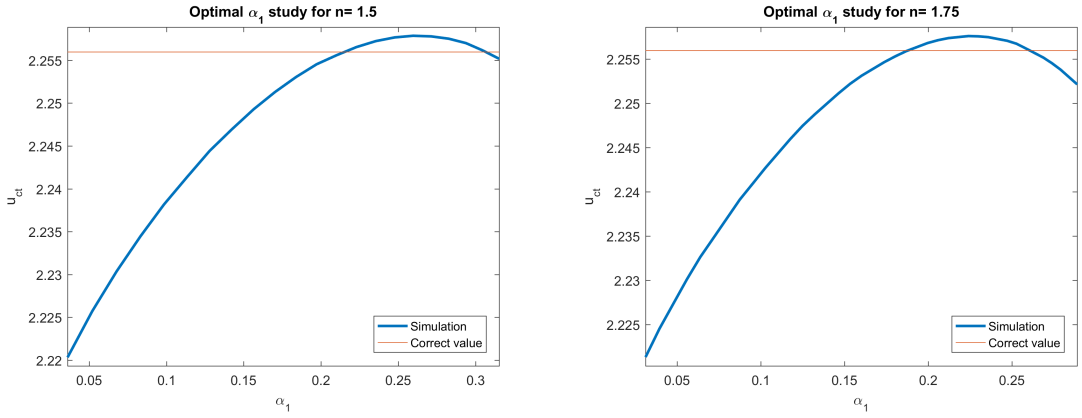


Figure 4.13 6E type 3 type 5 alternative (Second Study).

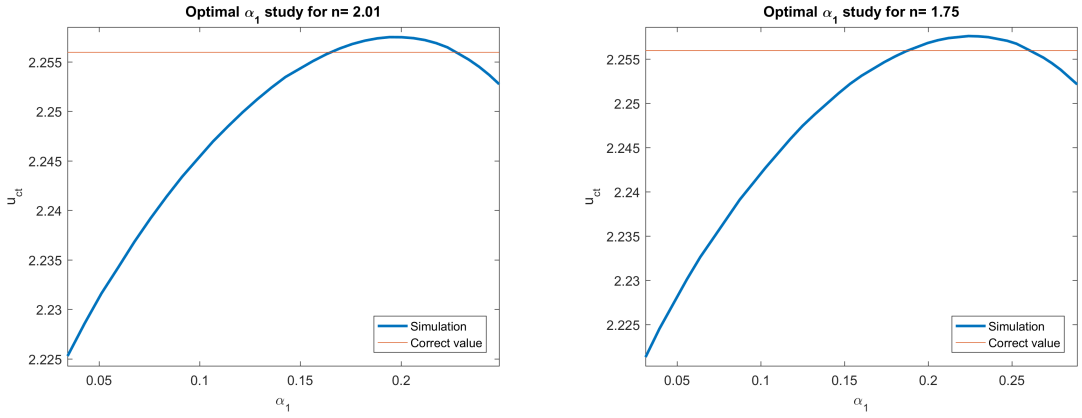


Figure 4.14 6E type 3 type 5 structured and alternative (Second Study).

4.1.7 6E type 2 type 0

The structured case suffers from the same phenomenon as those before: for high α_1 values the curve quickly decreases. Using both a structured mesh and an alternative process has more or less the same effect as seen before: the cut-off point becomes evident although the result quickly diverges (fourth figure).

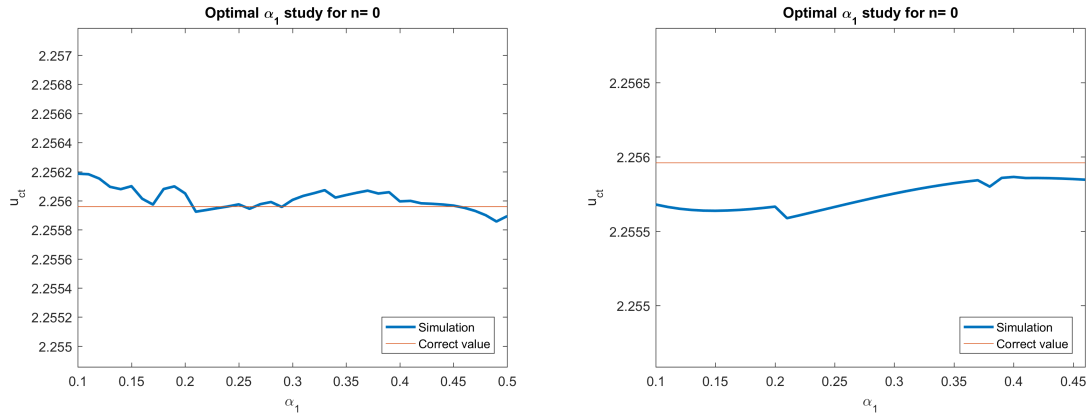


Figure 4.15 6E type 2 type 0 free and structured (Second Study).

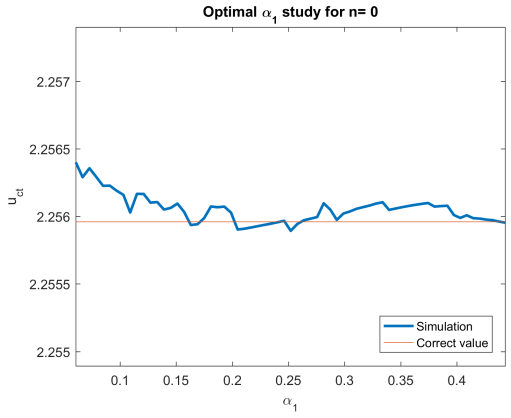


Figure 4.16 6E type 2 type 0 alternative (Second Study).

4.1.8 Analysis of results

As we have seen, the optimal would be a flat smooth curve with the highest possible range of α_1: flat to make the result be always close to the desired solution, and smooth to find out the best values for α_1 (since the cut-off point would then be clear). Those two traits seem to be at odds: the flattest the curve, the less smooth it becomes. This is specially evident when a cases and b cases are compared: a cases have clear cut-off points while b cases have an overall higher proximity to the curve. Besides, it is convenient to use m values between 2 and 2.25, since smaller values result in invalid cases (with the second circle inside the first) and bigger values result in poorer transitions. In any case, higher values still result in valid cases.
As for the influence of m on cut-off points, the first one (usually close to 0.2) seems to be less affected than the second one (usually close to 0.3), since the first one's range is around 0.04 while the second one's is around 0.1. This would imply a preference towards the first one, but we will keep using both.
Regarding the possible ways to improve curves, structured configurations may need a different adjustment since sometimes they do not reach the exact solution. The alternative method makes the curve less flat, but it diverges soon from the desired result (probably because the exact value is more easily inferred in one way or the other due to the effect of the numerical error in the adjustment). However, cut-off points with the correct solution, seem to remain close regardless of the method (either normal or alternative). Hence, using the alternative α_{1opt} value with the normal process may be a valid approach.
Regarding the choice of the cases, it has been proven that the case with 8 elements still have a regular behavior compared to its counterparts. Besides, 6 E type 3 type 5 b (third-higher scorer in its list) has been proven to be flatter than the higher scorer, who has also needed higher m values in comparison to others.A table with the results is added here. Flatness is measured as the α_1 range divided by the difference between maximum and minimum value.

Table 4.1 Second study summary.

	m	*Cut-off* α_1 *1*	*Cut-off* α_1 *2*	*max value*	*min value*	*flatness*
6E type 2 type 5 a free	1.75	n.a	n.a	2.27747	2.23811	10.92581
	2	0.23667	0.38913	2.25916	2.23628	17.47622
	2.25	0.22370	0.34219	2.25832	2.23870	18.85445
	2.5	0.21272	0.30543	2.25773	2.24328	22.84824
4E type 1 type 0 a free	n.a.	0.25303	0.42981	2.25626	2.25579	973.81879
4E type 1 type 0 a str	n.a.	n.a.	n.a.	2.25485	2.25184	149.62348
4E type 1 type 0 a alt	n.a.	0.24103	0.42942	2.25607	2.25568	1194.75133
4E type 1 type 5 b free	2.5	unstable	0.33979	2.25677	2.25600	424.61468
	3	n.a.	0.22892	2.25669	2.25591	356.46915
4E type 1 type 5 b str	2.5	n.a.	n.a.	2.25674	2.25603	467.40115
	3	0.24988	0.24988	2.25671	2.24713	29.25098
4E type 1 type 5 b alt	1.5	n.a.	n.a.	2.25719	2.25587	210.90751
	1.75	0.18464	nob	2.25719	2.25587	210.90751
6E type 5 type 2 a	1.75	0.20239	ob	2.26054	2.25029	39.02325
	2	0.19148	0.37761	2.25948	2.24392	25.70539
	2.25	0.18381	0.33279	2.25879	2.24519	27.20429
	2.5	0.17226	0.29780	2.25828	2.24639	27.74568
8E type 2 type 4 a free	1.75	0.20677	ob	2.26080	2.25032	38.16759
	2	0.19418	0.37823	2.25975	2.24228	22.89503
	2.25	0.18474	0.33255	2.25903	2.24369	24.11866
	2.5	0.17429	0.29682	2.25846	2.24516	24.81408
8E type 2 type 4 a str	1.75	0.20738	0.44111	2.26092	2.25527	70.67593
	2	0.19443	0.38060	2.25990	2.24215	22.54408
	2.25	0.18290	0.33478	2.25917	2.24369	23.90680
	2.5	0.17251	0.29904	2.25862	2.24517	24.52088
6E type 3 type 5 b free	1.75	0.19267	0.37745	2.25608	2.25591	2322.47896
	2	0.19488	0.25193	2.25606	2.25573	1236.60949
	2.25	0.15498	0.21069	2.25607	2.25576	1222.45108
	2.5	0.14247	0.14247	2.25601	2.25578	1492.75474
6E type 3 type 5 b str	1.75	0.13859	0.26178	2.25605	2.25585	2032.04703
	2	0.11753	0.24192	2.25599	2.25332	149.93054
	2.25	0.11197	0.38549	2.25600	2.25589	3318.65461
	2.5	n.a.	n.a.	2.25593	2.25588	5833.46820
6E type 3 type 5 b alt	1.5	0.21500	0.30677	2.25786	2.22033	8.52645
	1.75	0.18769	0.26072	2.25759	2.22133	8.27192
	2	0.16505	0.22744	2.25749	2.22529	8.69478
6E type 2 type 0 free	n.a.	0.23957	0.29161	2.25619	2.25592	1913.19397
6E type 2 type 0 str	n.a.	n.a.	n.a.	2.25587	2.25559	1621.14740
6E type 2 type 0 alt	n.a.	0.24093	0.26011	2.25640	2.25589	886.75745

ob: out of boundaries **nob: nearly out of boundaries** **n.a.: not an answer**

5 Third Parametric Study: optimal Singular Element-Crack proportion

5.1 Description

Next step should be to analyze the effect of the hexagon side-length. All the previous work may not be useful if the exact solution diverges too soon after a change in the hexagonal shape around. Hence, we need to verify that the results stay in a certain interval, close to the exact solution, for enough side-length values. In case that the solution quickly diverges, study two should be repeated for a higher side-length value in order to compare the results again.
This study intends to find the range of valid α_1 values, this is, the size of the singular element in proportion to the crack length (expressed as a percentage). We will be plotting the analysis solution alongside the exact solution, employing: the best hexagon-α_1 proportion and the best α_1-α_2-α_3 proportion. The band of $\pm 1\,\%$ error in the solution is also plotted in order to give an idea of approximated values. This is, there will be three straight lines: the solution, 1.01 times the solution and 0.99 times the solution. In any case, a different configuration might make the interval wider.
For every case, the optimal α_1 has been taken from the previous study. We assume $m = 2.25$ for two crowns and $m = 1.75$ for one crown (except when indicated otherwise) although a slightly lower value could be more representative.
Finally, a table of results is presented in which we sum up the validity range and the recommended size of the elements.

5.1.1 6E type 2 type 5 a free

As seen here, the general behaviour will be the following: the exact solution is achieved for the hexagon side-length used until now, and for smaller values it remains very close (sometimes it even converges again, a logical behavior taking into account that the element size is approaching zero). For higher values (usually around $18\,\%$ of the crack length), the solution quickly diverges.
Here, more specifically, there are two graphics: one uses the smallest α_1 obtained before, known as #1, and the other uses the highest α_1, known as #2. Both are very similar, but this will not always be the case.

5.1.2 4E type 1 type 0 a

In both cases, the phenomenon of convergence for very little α_1 values is observed. All the graphics are basically the same except for the first one, whose validity region end was close to the chosen α_1, and hence, seems to be influenced by it. This error, as seen before, is fixed by the alternative method, which gives a similar result to the others considered correct.
In comparison to the previous one, although this one converges for small values, it also reaches values more distant to the solution for more α_1. Thus we will be looking for cases who remain closer to the exact solution (that is, again, a flatter curve).

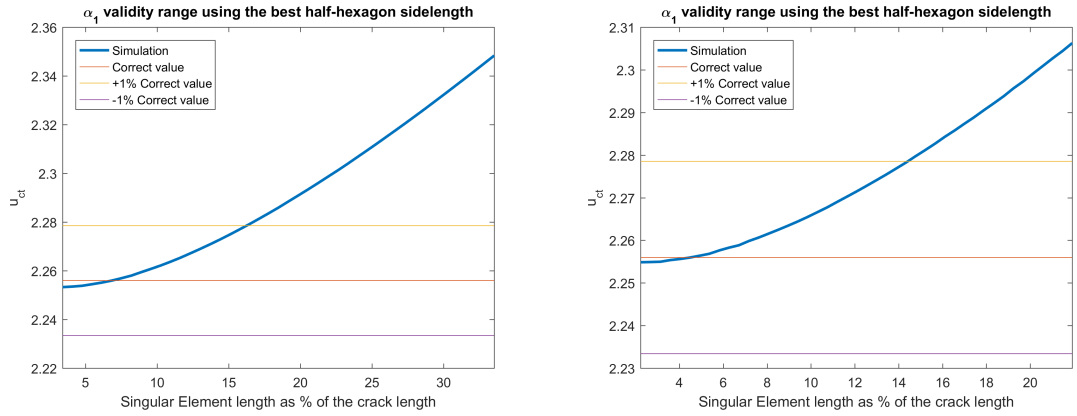


Figure 5.1 6E type 2 type 5 a free.

free

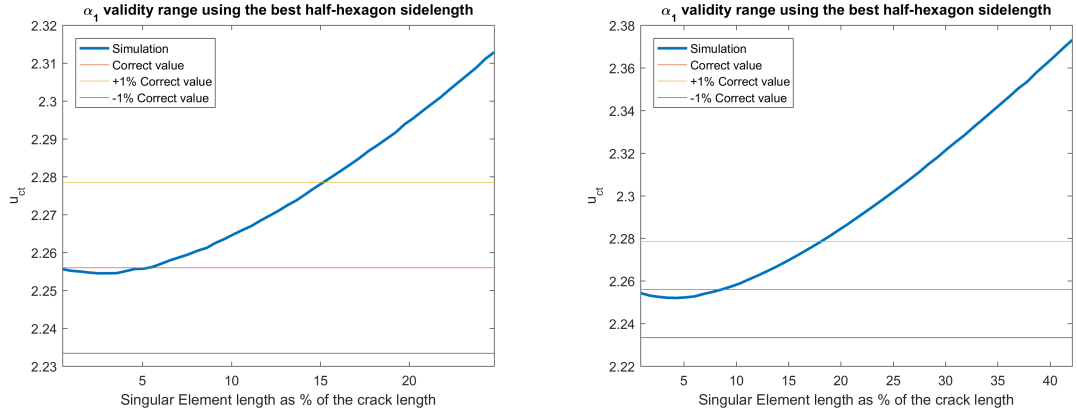


Figure 5.2 4E type 1 type 0 a free (Second Study).

alternative

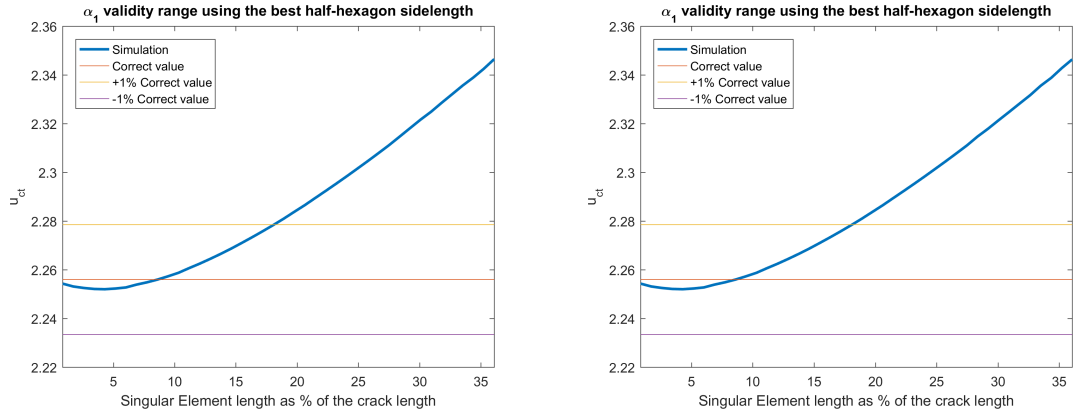
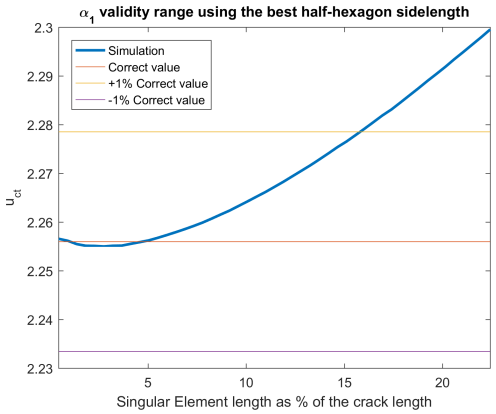


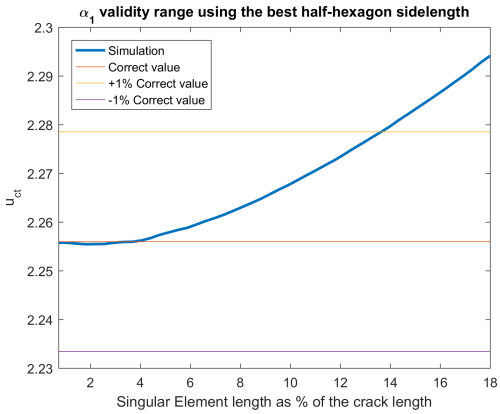
Figure 5.3 4E type 1 type 0 a alternative (Second Study).

5.1.3 4E type 1 type 5 b

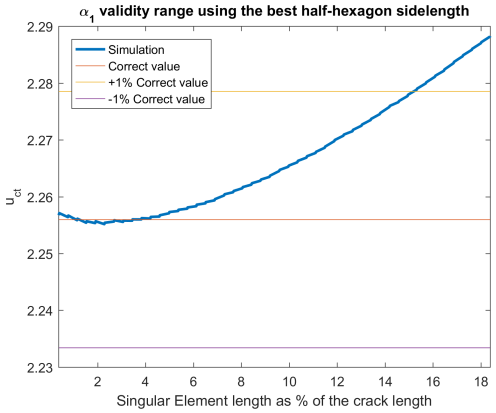
Conclusions are similar to the previous ones, this time every graphic seems to coincide without major issues (although the results do not perfectly coincide). The lower image seems to have a tooth saw pattern, but this is due to the sampling, which is much higher in the alternative process, and makes slight changes in the curve more evident. Due to this case's limitations, $m = 2.5$.



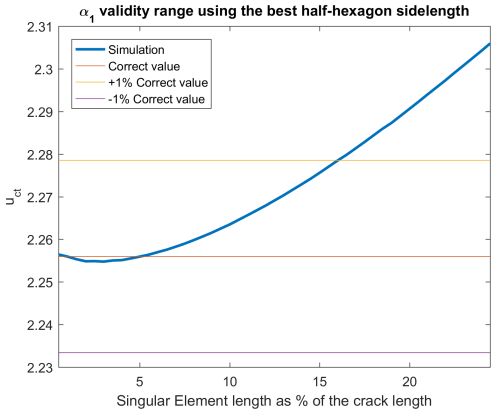
(a) Free inside-hexagon meshing.



(b) 6E type 5 type 2 free.



(a) Alternative process.



(b) Structured inside-hexagon meshing.

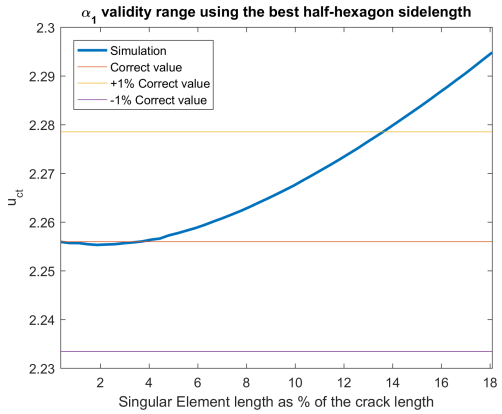
5.1.4 6E type 5 type 2 a

Result shown in the previous image.

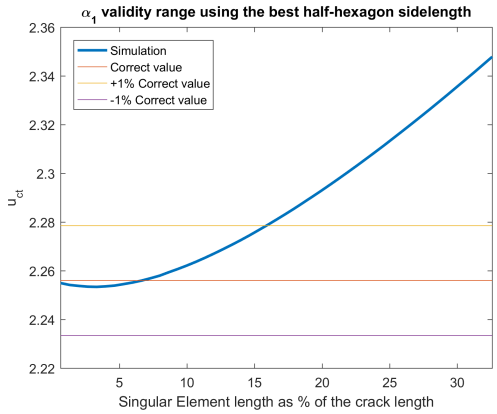
5.1.5 8E type 2 type 4 a

In this case both graphics have noticeable differences because the meshing is radically different in both cases. But, as a confirmation that this case was not exceptionally favorable (remember that this case was included to confirm that this case was still valid and not too deformed), has one of the narrowest ranges (for the free case).

Free

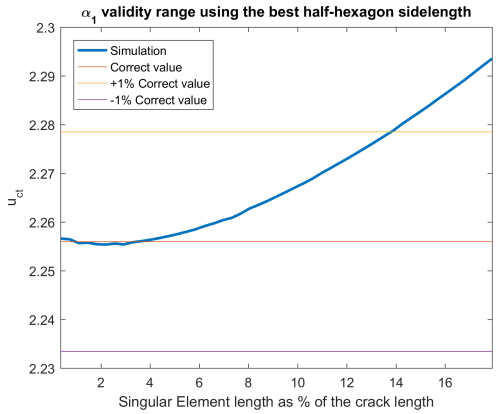


(a) Use of #1 α_1 for the free case.

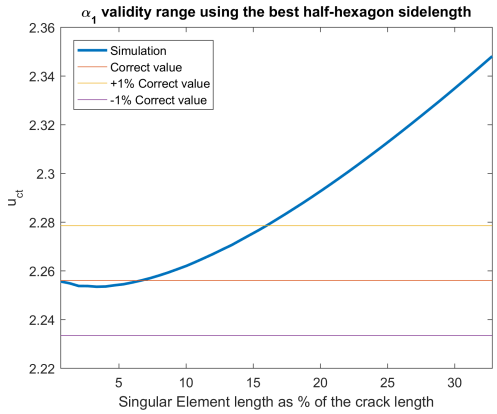


(b) Use of #2 α_1 for the free case.

Structured



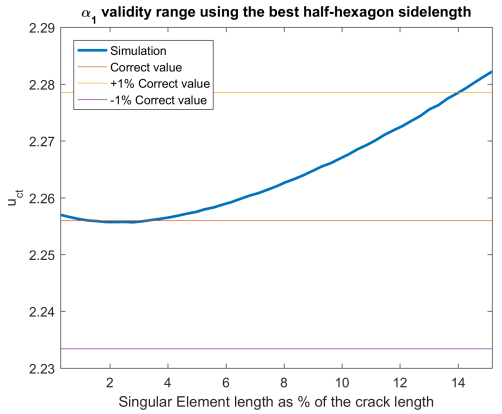
(a) Use of #1 α_1 for the structured case.



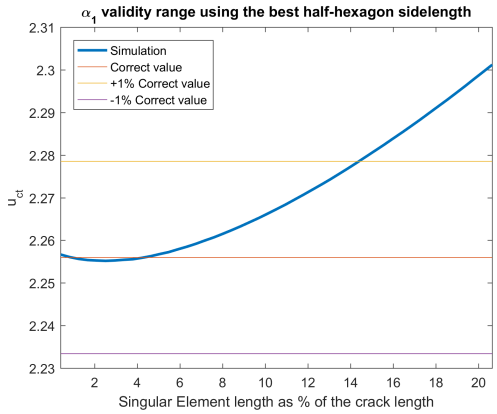
(b) Use of #2 α_1 for the structured case.

5.1.6 6E type 3 type 5 b

Free

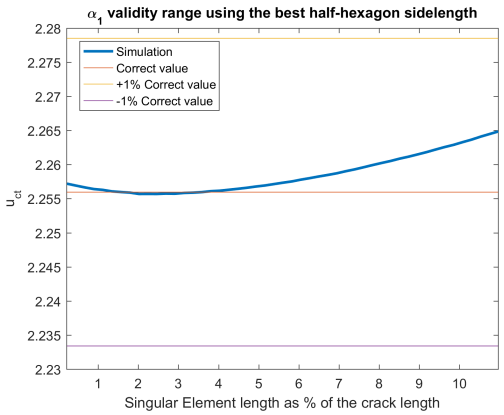


(a) Use of #1 α_1 for the free case.

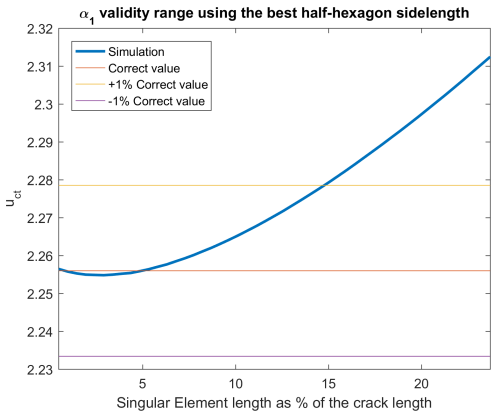


(b) Use of #2 α_1 for the structured case.

Structured



(a) Use of #1 α_1 for the free case.



(b) Use of #2 α_1 for the structured case.

Alternative

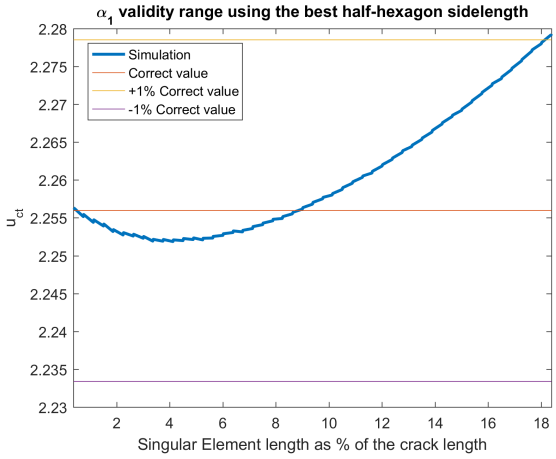


Figure 5.10 6E type 3 type 5 b alternative process.

In this case it is more relevant how structured meshing seems to flatten the curve, an effect not that evident until now. However, this comes at a cost: for small α_1 values, the limitations of out configuration prevent the element to grow up to its limit (as seen in the third image). Although this improves the results, the added difficulty does not seem to make up for the result. As a test, the results are here presented for $m = 2$ for two crowns and $m = 1.5$ for one crown. This does not change the habitual range of validity.

5.1.7 6E type 2 type 0 b

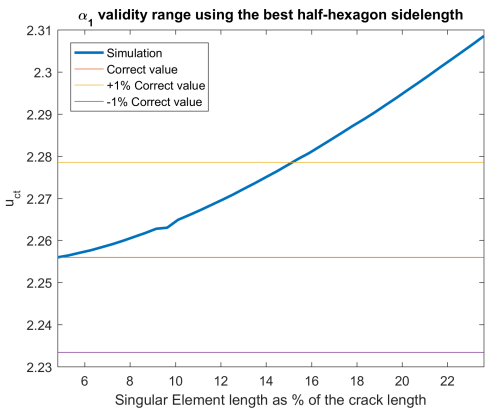
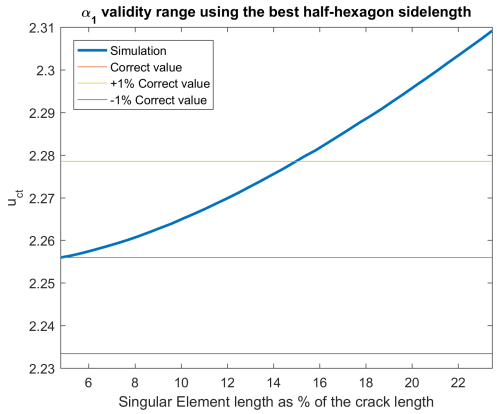
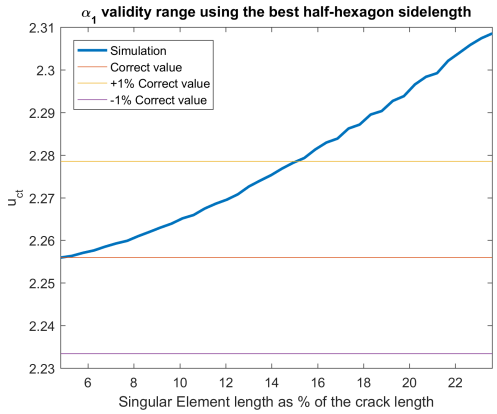


Figure 5.12 Structured.

In this case, the cut-off point is close to the start of the study. It is worth noting that the structured case was invalid in the beginning. no α_1 could meet the desired value with the hexagon side-length of 0.1. However, now it is possible to achieve by slightly changing that value. There are a few inconveniences in structured meshes. A slight miscalculation in the proportions may make the curve loose its parabolic shape, which makes only one cut-off point to appear, as it will be shown in the next page. Results are summed in the table below, with the



(a) Free.



(b) Alternative.

percentage indicating percentage of the crack length.

Table 5.1 Third study summary table.

	α_1	*Best Singular Element Size (%)*	*Max α_1 within 1% accuracy (%)*
6E type 2 type 5 free	0.22369874	4.48928736	14.4165444
	0.34219	6.84384561	16.2561366
4E type 1 type 0 free	0.25303	5.3787546	15.1773706
	0.42981	8.59627852	18.1121094
4E type 1 type 0 alt	0.24103	8.59740526	18.1131118
	0.42942	8.58832105	18.1058862
4E type 1 type 5 free	0.33979	4.71241692	15.8042594
4E type 1 type 5 str	0.24988	5.02865919	16.0126245
4E type 1 type 5 alt	0.18464478	3.82144632	15.28741
6E type 5 type 2 free	0.18381	3.71832262	13.6458931
8E type 2 type 4 free	0.18474	3.64200668	13.6136358
	0.33255	6.65096916	15.882028
	0.18290	3.42018662	13.8166996
	0.33478	6.6273513	15.9984622
6E type 3 type 5 free	0.15498	3.27773902	14.0275329
	0.21069	4.30452638	14.3911798
6E type 3 type 5 str	0.11197	3.57832243	12.6156338
	0.24192	4.9382774	14.7730228
6E type 3 type 5 alt	0.18769	8.65937038	18.05977937
6E type 2 type 0 free	0.23957	4.79091461	14.9687985
6E type 2 type 0 str	0.24093	4.77642037	15.1484462
6E type 2 type 0 alt	0.24093	4.79091461	15.0964382

5.2 Table of results

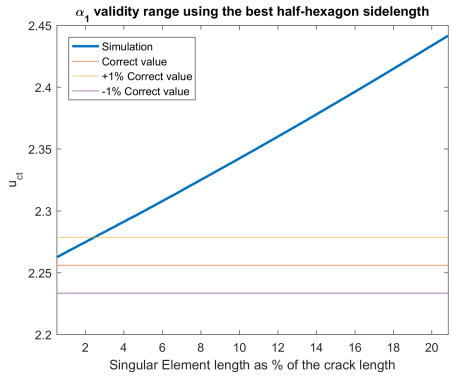


Figure 5.13 Poorly adjusted structured case, in which the parabolic shape is lost alongside its benefits.

In conclusion, the results are within a 1% accuracy range when the singular element is less than 16% of the crack length. The results are practically accurate when the element is around 5% of the crack length. This, although a good result, is slightly worse than results put forward by [19], in which a quarter point of 20% of the crack length can still be acceptable. This, however, might be achieved by using a natural isoparametric element and not a collapsed one, as the previous results were found employing these.

It is important to note that the highest-scoring configuration (4E type 1 type 0) also has the widest range of validity, and coincidentally, remains close enough to the solution for the most part for small α_1. The narrower interval is due to structured meshes for 6E type 3 type 5. The flatter curves are those of 4E type 3 type 5 (which even has two valid cut-off points), although for the structured mesh it has a range limitation. They are followed by 6E type 2 type 5. In general, it will be advisable to keep the element under 10% of the crack length and use the aforementioned configurations.

6 Conclusions, summary, further research

6.1 Conclusions

As we have seen, we can conclude that the element can properly capture the displacement tendencies near the crack tip, but to ensure a correct solution, the following recommendations should be applied:

- The element should preferably measure between 4 % and 8 % of the crack length, but a 15 % of the crack length still gives solution with 1 % accuracy.
- One or two crowns of non-singular elements should surround the singular elements.
- The first crown size should be between 0.65 and 0.7 times the singular element size.
- The second crown size should be between 0.88 and 1 times the first crown size.
- The transitioning half-hexagon can determine the solution and its length is suggested to be 10 % of the crack length.
- A crown configuration of the types summarised below can improve the performance of the element.

The following table presents the suggested: configurations, α_i, hexagon side-length, element size, adjustment coefficients and validity intervals.

6.1.1 Developed materials

Handbook rules aside, during the research process, the following tools have been created:

- A MATLAB script to apply boundary conditions of any kind to a square plate, a MATLAB script to read Abaqus .inp files and carry a parametric study by comparing the result of several of them.
- a MATLAB script to visually present the results of such study and find an adjustment for the right element and crown(s) proportion.
- A web-page to visualize meshes and visually pick nodes from it (to make sure your boundary. conditions are being applied to the desired nodes)
- More than 350 Python macros to parametrically build meshes (and the MATLAB scripts to easily change the defining commands).
- More than 214023 meshes studied
- And several MATLAB scripts to refine the previous conclusions (finding the right hexagon-SE proportion or the right SE-crack proportion)

6.2 Summary

6.2.1 Two crown cases

Firstly, α_1 is chosen. Then, α_3 is chosen as $\alpha_3 = m\alpha_1$ (for instance, $m = 2$). Then, α_2 is calculated by the following adjustment:

$$\alpha_2 = p_{00} + p_{10}\alpha_1 + p_{01}\alpha_3 + p_{20}\alpha_1^2 + p_{11}\alpha_1\alpha_3 + p_{02}\alpha_3^2$$

And in order to simplify the resulting expression:

$$\alpha_2 = p_{00} + (p_{10} + mp_{01})\alpha_1 + (p_{20} + mp_{11} + m^2 p_{02})\alpha_1^2$$

$$\alpha_2 = p_0 + p_1\alpha_1 + p_2\alpha_1^2$$

6.2.2 One crown cases

In these cases, α_2 is fixed and α_1 is calculated as follows. Evidently: $p_{00} = p_0$, $p_{10} = p_1$, $p_{20} = p_2$.

$$\alpha_1 = p_{00} + p_{10}\alpha_2 + p_{20}\alpha_2^2$$

6.2.3 Inverse method

As seen before, choosing α_2 beforehand reduces oscillations in the result. Hence, $\alpha_3 = m_2\alpha_2$ (for instance $m_2 = 1.75$). And finally, $\alpha_1 = p_{00} + p_{10}\alpha_2 + p_{01}\alpha_3 + p_{20}\alpha_2^2 + p_{11}\alpha_2\alpha_3 + p_{02}\alpha_3^2$, which can be similarly expressed as: $\alpha_1 = p_{00} + (p_{10} + m_2 p_{01})\alpha_2 + (p_{20} + m_2 p_{11} + m_2^2 p_{02})\alpha_2^2$ or $\alpha_1 = p_0 + p_1\alpha_2 + p_2\alpha_2^2$
The same thing applies to one-crown cases just deleting the square term.

6.2.4 Table of results

The following parameters are presented in the table on the following page:

- SE size: recommended size of the singular element near the crack tip (measured from the crack vertex to its base).
- hex. side: hexagon side-length used in the calculations.
- α_1, α_2, α_3: their optimal value.
- p_{ij}: adjustment coefficients.
- $\alpha_2 - \alpha_1$: size of the first crown divided by the hexagon side-length (like α coefficients are expressed).
- $\frac{\alpha_2 - \alpha_1}{\alpha_1}$: size of the first crown divided by the singular-element size. Here we can see that the transition element should measure roughly 0.68 times the SE.
- $\frac{\alpha_3 - \alpha_2}{\alpha_2 - \alpha_1}$: size of the second crown divided by the first crown size.
- p_1, p_2, p_3: coefficients of the adjustment dependant only on one parameter.
- 1% acc. interval: upper limit of the 1% accuracy interval.

An Excel version has been created. It is interactive and allows the user to change m and m_2 values. This table is presented for appropriate values $m = 2.25$ and $m_2 = 1.5$.
The same nomenclature for coefficients is employed in each section. However, as explained here, they multiply different variables. Also, the symbol Idem is here employed to remark that the value is equal to the one in the upper box. Furthermore, an Excel file is attached to consult the values with all their decimals. A square is added to modify the m value, hence allowing to change the proportion between α values (depending on the case).
Additionally, the mutual validity of normal and alternative processes is confirmed, since their results are similar enough as seen in the table.

Table 6.1 Summary.

TWO-CROWN CASES																			
NORMAL PROCESS	**SE size**	**hex. side**	α_1	p_{00}	p_{10}	p_{01}	p_{20}	p_{11}	p_{02}	α_2	α_3	$\alpha_2-\alpha_1$	$\frac{\alpha_2-\alpha_1}{\alpha_1}$	$\alpha_3-\alpha_2$	$\frac{\alpha_3-\alpha_2}{\alpha_2-\alpha_1}$	p_0	p_1	p_2	**1 % acc. interval**
6E type 2 type 5 a free	4.4893	0.1	0.2237	0.0920	1.4444	-0.2205	0.0554	0.3029	0.1207	0.3716	0.5033	0.1479	0.6612	0.1317	0.8905	0.0920	0.9483	1.3478	14.4165
	6.8438	0.1	0.3422	Idem	Idem	Idem	Idem	Idem	Idem	0.5744	0.7699	0.2322	0.6785	0.1956	0.8424	Idem	Idem	Idem	16.2561
4E type 1 type 5 b free	4.7124	0.1	0.3398	-0.0017	1.6294	0.0038	0.0871	0.0329	-0.0091	0.5681	0.7645	0.2284	0.6720	0.1964	0.8600	-0.0017	1.6379	0.1152	15.8043
4E type 1 type 5 b str	5.0287	0.1	0.2499	Idem	Idem	Idem	Idem	Idem	Idem	0.4148	0.5622	0.1649	0.6599	0.1475	0.8943	Idem	Idem	Idem	16.0126
6E type 5 type 2 a free	3.7183	0.1	0.1838	0.0537	1.4840	-0.1386	0.0837	0.2403	0.0809	0.3041	0.4136	0.1203	0.6543	0.1095	0.9104	0.0537	1.1721	1.0340	13.6459
8E type 2 type 4 a free	3.6420	0.1	0.1847	0.0600	1.4878	-0.1609	0.0803	0.2354	0.1012	0.3063	0.4157	0.1215	0.6580	0.1094	0.8998	0.0600	1.1259	1.1223	13.6136
	6.6510	0.1	0.3326	Idem	Idem	Idem	Idem	Idem	Idem	0.5585	0.7482	0.2260	0.6795	0.1897	0.8395	Idem	Idem	Idem	15.8820
8E type 2 type 4 a str	3.4202	0.1	0.1829	Idem	Idem	Idem	Idem	Idem	Idem	0.3035	0.4115	0.1206	0.6592	0.1081	0.8964	Idem	Idem	Idem	13.8167
	6.6274	0.1	0.3348	Idem	Idem	Idem	Idem	Idem	Idem	0.5627	0.7533	0.2279	0.6808	0.1906	0.8360	Idem	Idem	Idem	15.9985
6E type 3 type 5 a free	3.2777	0.1	0.1550	0.0014	1.6216	0.0004	0.1420	0.0173	-0.0042	0.2567	0.3487	0.1017	0.6565	0.0920	0.9040	0.0014	1.6226	0.1596	14.0275
	4.3045	0.1	0.2107	Idem	Idem	Idem	Idem	Idem	Idem	0.3504	0.4741	0.1397	0.6630	0.1237	0.8855	Idem	Idem	Idem	14.3912
6E type 3 type 5 a str	3.5783	0.1	0.1120	Idem	Idem	Idem	Idem	Idem	Idem	0.1851	0.2519	0.0731	0.6532	0.0668	0.9138	Idem	Idem	Idem	12.6156
	4.9383	0.1	0.2419	Idem	Idem	Idem	Idem	Idem	Idem	0.4033	0.5443	0.1614	0.6671	0.1410	0.8739	Idem	Idem	Idem	14.7730
ALTERNATIVE PROCESS	**SE size**	**hex. side**	α_1	p_{00}	p_{10}	p_{01}	p_{20}	p_{11}	p_{02}	α_2	α_3	$\alpha_2-\alpha_1$	$\frac{\alpha_2-\alpha_1}{\alpha_1}$	$\alpha_3-\alpha_2$	$\frac{\alpha_3-\alpha_2}{\alpha_2-\alpha_1}$	p_0	p_1	p_2	**1 % acc. interval**
4E type 1 type 5 b alt	3.8214	0.1	0.1846	0.0013	0.6120	-0.0024	-0.0174	-0.0113	0.0053	0.3066	0.4599	0.1220	0.6605	0.1533	1.2570	0.0013	0.6084	-0.0344	15.2874
6E type 3 type 5 b alt	8.6594	0.1	0.1877	-0.0569	0.6245	0.1714	-0.0071	-0.0406	-0.1201	0.3157	0.4735	0.1280	0.6819	0.1578	1.2332	-0.0569	0.8815	-0.3383	18.0598
ONE-CROWN CASES																			
NORMAL PROCESS	**SE size**	**hex. side**	α_1	p_{00}	p_{10}	p_{01}	p_{20}	p_{11}	p_{02}	α_2	α_3	$\alpha_2-\alpha_1$	$\frac{\alpha_2-\alpha_1}{\alpha_1}$	$\alpha_3-\alpha_2$	$\frac{\alpha_3-\alpha_2}{\alpha_2-\alpha_1}$	p_0	p_1	p_2	**1 % acc. interval**
4E type 1 type 0 ab free	5.3788	0.1	0.2530	0.0009	0.6179	0.0000	0.1480	0.0000	0.0000	0.4197	n.a.	0.1667	0.6588	n.a.	n.a.	0.0009	0.6179	0.1480	15.1774
	8.5963	0.1	0.4298	Idem	Idem	Idem	Idem	Idem	Idem	0.7236	n.a.	0.2938	0.6836	n.a.	n.a.	Idem	Idem	Idem	18.1121
6E type 2 type 0 b free	4.7909	0.1	0.2396	0.0009	0.6256	0.0000	0.1475	0.0000	0.0000	0.3988	n.a.	0.1592	0.6646	n.a.	n.a.	0.0009	0.6256	0.1475	14.9688
6E type 2 type 0 b str	4.7764	0.1	0.2409	Idem	Idem	Idem	Idem	Idem	Idem	0.4011	n.a.	0.1602	0.6647	n.a.	n.a.	Idem	Idem	Idem	15.1484
ALTERNATIVE PROCESS	**SE size**	**hex. side**	α_1	p_{00}	p_{10}	p_{01}	p_{20}	p_{11}	p_{02}	α_2	α_3	$\alpha_2-\alpha_1$	$\frac{\alpha_2-\alpha_1}{\alpha_1}$	$\alpha_3-\alpha_2$	$\frac{\alpha_3-\alpha_2}{\alpha_2-\alpha_1}$	p_0	p_1	p_2	**1 % acc. interval**
4E type 1 type 0 ab alt	8.5974	0.1	0.2410	-0.0003	0.6162	0.0000	-0.0301	0.0000	0.0000	0.3995	n.a.	0.1584	0.6573	n.a.	n.a.	-0.0003	0.6162	-0.0301	18.1131
	8.5883	0.1	0.4294	Idem	Idem	Idem	Idem	Idem	Idem	0.7229	n.a.	0.2935	0.6835	n.a.	n.a.	-0.0003	0.6162	-0.0301	18.1059
6E type 2 type 0 b alt	4.7909	0.1	0.2409	-0.0004	0.6137	0.0000	-0.0301	0.0000	0.0000	0.4011	n.a.	0.1602	0.6648	n.a.	n.a.	-0.0004	0.6137	-0.0301	15.0964

6.3 Further research

The next steps regarding the research are suggested to be the following:

- Convergence keeping the suggestions while augmenting the number of elements in the plate. For this, special care to the relation between the meshing parts and the size of the element should be given. With this, an improvement for small SE values could be achieved.
- Study on the effect of transition region. As seen before, structured meshing affects the optimal values but can make the behaviour versus SE size more manageable. Finding the best way to transition (fixating the number of elements on the half-hexagon and its arrangement) may make the study results more conclusive.
- Study on the effect of using singular-elements as transition elements in the crown. As referenced before, the dominance zone is neither singular or non-singular, hence using linear or quadratic elements may help capture the displacements, specially for high-precision studies with fine meshes.
- Changing the element to give a quadratic behaviour on the opposite side instead of a linear one.
- Implementing an Abaqus sub-routine to use the element directly on Abaqus, in order to implement this whole study method in one program, benefiting from its time-cost-improved algorithm.
- Using an isoparametric element instead of a collapsed one, revisiting the element equations but trying to build it in another way.

6.4 Personal conclusions

Thanks to this project, I feel I have acquired several new abilities. For instance, planning has proven to be a major issue, having to employ several computers at a time to reduce total time computing cases, or leaving the computer calculating during the night instead of working simultaneously on two things while keeping an eye on simulations. Furthermore, while managing a gargantuan amount of information, I learnt how important it is to keep a code for identification and sticking to it. Similarly, I learnt that using a lot of information also require useful tools to analyse it. Working on and developing such tools is fundamental and should not be avoided by trying to re-adapt other tools or doing the work manually.
Finally, I learnt the importance of revising the results and keeping up to date the final product (in this case, this document). Losing the main purpose of the study (putting forward conclusions) can lead to self-indulgence in minor details. That is, perfectionism is fine as long as it does not make you lose time and forget your goals. All in all, it is not only the amount of work put into a research, but also the way everything is explained and the need of those results.

7 Appendix: Auxiliary code

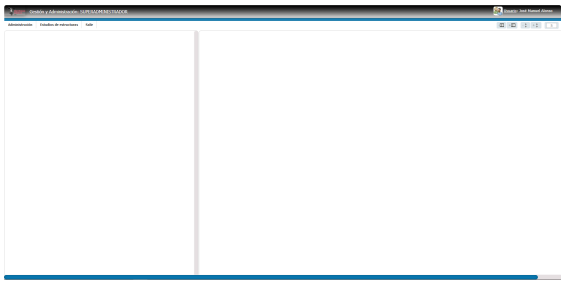
During the preparation of the previous studies, a graphical tool to visually select nodes in the meshes was developed. This was programmed in order to make sure that the lists of nodes generated by our code, which automatically imposes boundary conditions, corresponds to the desired selection of nodes. Although it is not necessary for the results, it is a tool that might be of help for future modifications of the code to adapt it to other cases. In the following sections, the tool's code and purpose is explained. Hyperlinks redirect to web pages in the digital edition. It was originally developed in Spanish, so text in the images (in Spanish) will be referred to in English (as the previous sections).

7.1 Introduction

In order to study the meshes, a little web application has been developped. This application is located in a server, a web hosting with the domain: www.ayrsoftsoluciones.com.
The app is run from a navigator window with the following url: http://www.ayrsoftsoluciones.com/nodos. This will redirect to the screen in the left.
To access, the username and password previously assigned by the administrator must be entered. It will display the right screen capture. There is a main menu with three options:

1. **Administration.** From this option we can register different users which will be able to use the app.
2. **Structural Study.** From this option we will work the different meshes.
3. **Exit.** To abandon the web application.

(a) Access screen.

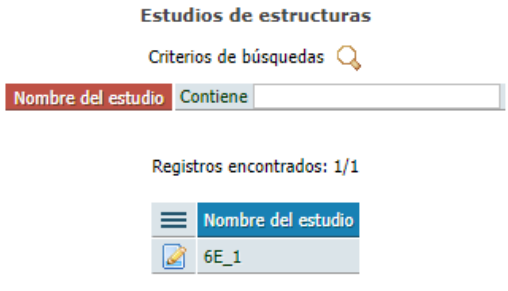


(b) First screen.

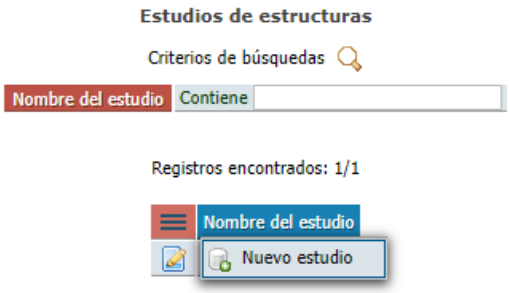
7.2 Application functioning

First of all, we must create a study. This study will have a name assigned and will have two archives associated containing information related to each mesh. When we click on Structural Study, we will find a screen like the left one.

In the left margin we can already observe a registered study, ‘6E_1’. To introduce a new study, for instance one called ‘TFG_study_example’. To do so, right click on the three lines beside the literal ‘Study name‘ and then select ‘New study’.

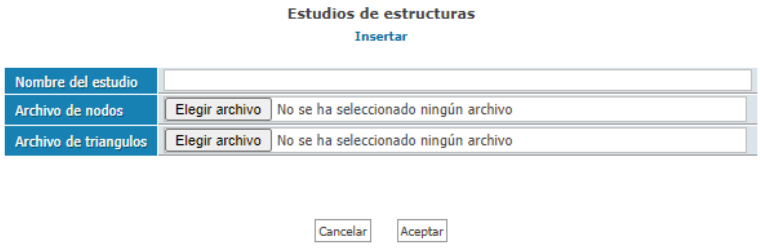


(a) Studies screen.

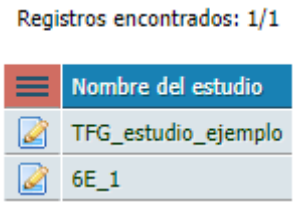


(b) Create a new study.

In the right frame a little form will be displayed. There we will write the study name and select two archives: the node archive (a ’.txt’file with the nodes coordinates, equivalent to the variable ‘coordenadas’), and the triangle archive (a ‘.txt’file with the nodal connection, equivalent to the variable ‘triangulos’). By clicking on the ‘accept’button, the left frame will display that the new study has been added to the list.



(a) Generating a new study.



(b) List of studies.

7.3 Actions to perform

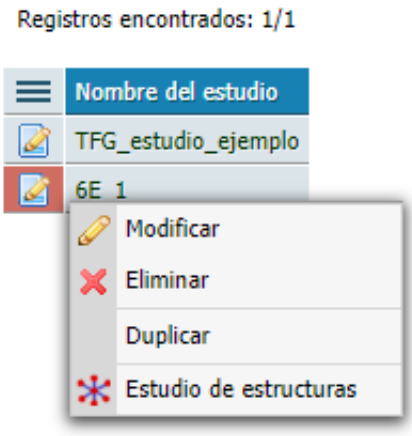


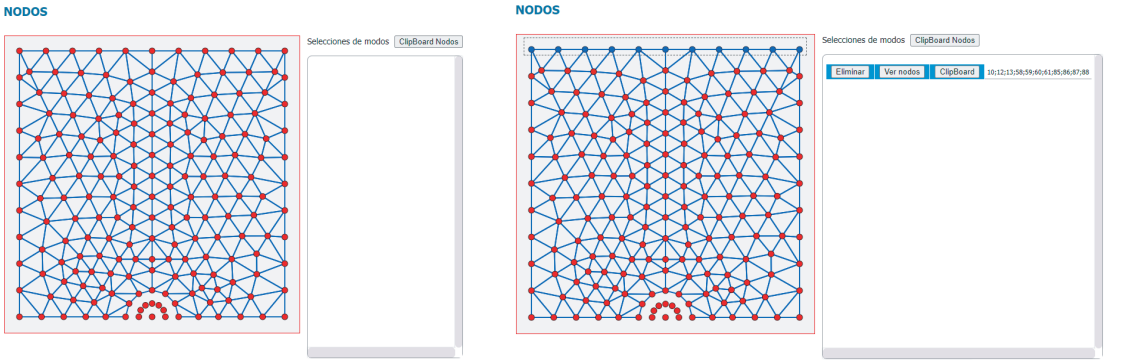
Figure 7.4 Plate and Reference System.

Taking as an example a study of a ’6E_type_1_type_-0’case. If we right-click on its row icon, a context menu will be displayed with the following options:

- **Modify:** If we want to change the study name or change the node or triangles archives.
- **Delete:** To permanently delete a case study.
- **Duplicate:** Clones a case study’s data.
- **Structural study:** Is what we are going to use to perform the study in the mesh defined by the archives. Clicking on this option a screen like this one will appear:

In the image we can see a mesh whose nodes and triangles have been created from the previous archives when registering the study. Nodes in red and triangle edges in blue.

At this point, the application allows us to drag our mouse over the nodes, selecting those who lay within the region that is delimited by the selecting rectangle, changing the node color to blue (to show such a state).



(a) Mesh display.

(b) Nodal selection.

When the mouse button is released, in the right frame of the figure, the correspondent number of the previously selected nodes is displayed. Repeating this process we can arrive to a determined quantity oflines with different lists of nodes.
Clicking on 'See node', in each moment we can see the set of nodes selected by that row.
If we click on 'Delete', we can take out the line of the desired set of nodes. If we click on 'Clipboard', the list of nodes (using its identification number) is copied to clipboard to paste them in our MATLAB script.

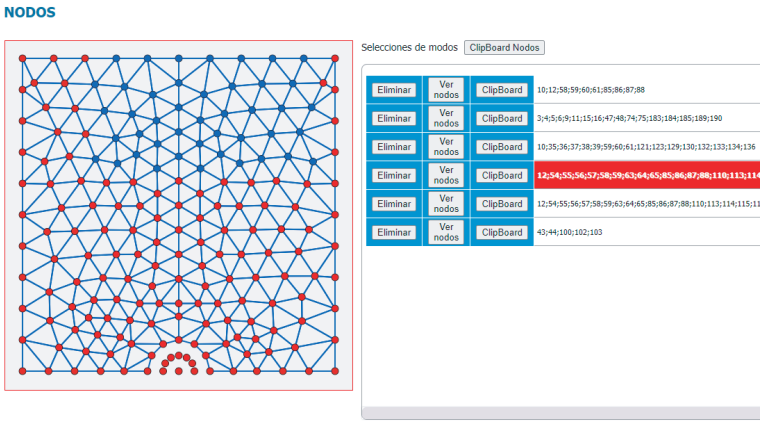


Figure Different lists display.

7.4 Technology

As it has been stated, the application is a web application hosted in the domain www.ayrsoftsoluciones.com, developed in PHP, using the database MySQL. The web page source code is here presented, alongside the JavaScript code that implements the vector graphic engine and event capture for the nodes selection.

7.4.1 PHP code

```
<?php

    define('PATH_GLOBAL',"../..");
    define('PATH_LOCAL',"..");

    // Includes
    include_once(" ../ access/
        ctrlsession .php");
    include_once(PATH_GLOBAL."/
        global.php");
    include_once(PATH_LOCAL."/
        global.php");

    include_once(
        PATH_GLOBALPHPCLASS."/
        AdoPhp.php");
    include_once(
        PATH_GLOBALPHPLIB."/
        vistas.php");
    include_once(PHPIDI."/
        stringsMsg_".$_SESSION["
        s_idioma"].".php");
```

Script 7.1 PHP Code.

Libraries inclusion.

```
    // Creates object command
    $cmd=new Comando($_SESSION[
        "s_cadenaconexion"]);
    if (!$cmd) die($TbMsg[2]); //
        BD connexion error
```

Creates an object command to read the database.

```
    $idestudio =$_GET["ide"];
```

Stores the identifying parameter of the study (the id).

```
    $rs=q_estudios( $idestudio );
    if (!$rs->EOF) die($TbMsg[10]);
        // Error: no registers
```

Creates a record set that reads the nodes and triangles archive from the database and proves its existence.

```
if (! file_exists (
    pathArchivosNodos."/". $rs->
    campos["archivonodos"])) die(
    "no existe archivo ".
    pathArchivosNodos."/". $rs->
    campos["archivonodos"]);
if (! file_exists (
    pathArchivosTriangulos."/".
    $rs->campos["
    archivotriangulos"])) die("no
     existe archivo ".
    pathArchivosTriangulos."/".
    $rs->campos["
    archivotriangulos"]);
```

Verifies the existence of archives in the configured route.

```
$svg="";
 $circles="";
 $lines="";
 $ids=1;
 $tbNodosX=array();
 $tbNodosY=array();
 $minx=0xFFF;
 $miny=0xFFF;
 $maxx=0;
 $maxy=0;
```

Variables set.

```
$fp = fopen(pathArchivosNodos."/
    ". $rs->campos["archivonodos"
    ], "r");
while (! feof ($fp)){
   $linea = fgets ($fp);
   $linea=preg_replace ([' /\s+/','
    /^\s|\s$/'],[' ',''], $linea
    );
    list ($x,$y,$z)=explode(" ",
     $linea);
   if ($x>$maxx) $maxx=$x;
   if ($x<$minx) $minx=$x;
   if ($y>$maxy) $maxy=$x;
   if ($y<$miny) $miny=$y;
}
fclose ($fp);
$LX=450;
$WX=500;
$DPLX=($WX-$LX)/2;
$wx=$maxx-$minx;
$wy=$maxy-$miny;
$factorx=$wx*$LX;
$factory=$wy*$LX;
$factor=$factorx;
if ($factor<$factory) $factor=
    $factory;
// die(" factor=$factor factor x
    =$factorx factory=$factory
     dplx=$wx dply=$wy //
    Maximo x=$maxx maximo y=
    $maxy minx=$minx miny=
    $miny");
$fp = fopen(pathArchivosNodos."/
    ". $rs->campos["archivonodos"
    ], "r");
while (! feof ($fp)){
   $linea = fgets ($fp);
   $linea=preg_replace ([' /\s+/','
    /^\s|\s$/'],[' ',''], $linea
    );
    list ($x,$y,$z)=explode(" ",
     $linea);
   $x=trim($x);
   $y=trim($y);
   $z=trim($z);
   $x*=$factor;
   $y*=$factor;
   $x+=$factor/2+$DPLX;
   $y=$LX-$y+$DPLX;
    $circles.='<circle id="'.$ids
    .'" r="5" stroke="black"
    class="nodo" fill="red" cx="
    '.$x.'" cy="'.$y.'" />';
   $tbNodosX[$ids]=$x;
   $tbNodosY[$ids]=$y;
   $ids++;
}
fclose ($fp);
$fp = fopen(
    pathArchivosTriangulos."/".
    $rs->campos["
    archivotriangulos"], "r");
while (! feof ($fp)){
   $linea = fgets ($fp);
   $linea=preg_replace ([' /\s+/','
    /^\s|\s$/'],[' ',''], $linea
    );
    list ($n1,$n2,$n3)=explode(" ",
     $linea);
   $n1=trim($n1);
   $n2=trim($n2);
   $n3=trim($n3);
   $x1=$tbNodosX[$n1];
   $y1=$tbNodosY[$n1];
   $x2=$tbNodosX[$n2];
   $y2=$tbNodosY[$n2];
   $x3=$tbNodosX[$n3];
   $y3=$tbNodosY[$n3];
    $lines.='<line x1="'.$x1.'" y1
    ="'.$y1.'" x2="'.$x2.'" y2="
    '.$y2.'" style="stroke:rgb
    (27,76,222); stroke-width:2"
    />';
    $lines.='<line x1="'.$x1.'" y1
    ="'.$y1.'" x2="'.$x3.'" y2="
    '.$y3.'" style="stroke:rgb
    (27,76,222); stroke-width:2"
    />';
    $lines.='<line x1="'.$x3.'" y1
    ="'.$y3.'" x2="'.$x2.'" y2="
    '.$y2.'" style="stroke:rgb
    (27,76,222); stroke-width:2"
    />';
}
fclose ($fp);
$svg=$lines. $circles;
?>
```

This reads the nodes and triangles archives and generates the html code needed for its graphic representation. Some ‘svg’ labels are used, which in html5 draw vectorial graphs. Graphics are not drawn as ‘jpg´ or ‘png´ but they depict ‘html´ labels of scalable graphics.

```
<!DOCTYPE html>
  <head>
    <meta name="keywords" content
     =""/>
    <meta http-equiv="Content-
     Type" content="text/html;
     charset=utf-8" />
    <meta name="viewport" content=
     "width=device-width, initial-
     scale=1"/>
    <!-- Estilos Css-->
    <link rel="shortcut icon" href
     =" ../../ styles/images/
     favicon.ico"/>
     <!--Javascript librerias -->
    <script src="https://ajax.
     googleapis.com/ajax/libs/
     jquery/3.3.1/jquery.min.js"
     ></script>
    <title>Gestión y administración</title>
    <meta http-equiv="Content-
     Language" content="es" />
    <link rel="stylesheet" href="
     ../../ styles/css/
     globalstyles.css?nocache="
     type="text/css" />
    <link rel="stylesheet" href="
     ../templates/default/css/
     cssdefault.css?nocache="
     type="text/css" />
    <script language="javascript"
     src="../ jscript/nodos.js"></
     script>
    <style>
```

Here the javascript code (added later) is included. It is a language that runs in the navigator. The previous is executed in the server, which generates an html code that runs in the client server.

```
      #workspace
      {
        background-color: #F5F1F2
      ;
        border: 1px solid red;
        width:<?php echo $WX?>;
        height:<?php echo $WX?>;
      }
      /* Selectioning rectangle */
       .selrecdiv {
        position: absolute;
        top:0px;
        left:0px;
        display:none;
        border:0.1em dashed #
      474747;
        margin:0;
        width:10px;
        height:10px;
        padding:0px;
      }
      .non-selectable {
         -moz-user-select: none;
         -webkit-user-select: none
      ;
         -ms-user-select: none;
```

```
            user-select: none;
            -khtml-user-select: none;
        }
        .contenedornodos{
          border-radius: 8px;
          border:1px solid #999999;
          height: 500px;
          overflow: scroll
        }
    </style>
</head>
<body>
<h1 style="margin:10px">NODOS
    </h1>
```

Here the nodes that have been selected by the user are stored.

```
<table cellspacing="10">
  <tr>
    <td valign="top">
      <div id="areatrabajo">
        <svg xmlns="http://
        www.w3.org/2000/svg"
          shape-rendering="
        geometricPrecision"
          viewBox="0 0 <?php
        echo $WX?> <?php echo $WX
        ?>"
          width=<?php echo $WX
        ?>
          height=<?php echo
        $WX?>
          fill="white"
          id="svgNodos"
          stroke="black"
          stroke-width="0.8px"
          style="background-
        color: #F5F1F2">
          <?php
            echo $svg;
          ?>
        </svg>
      </div>
    </td>
    <td valign="top" class="non
    -selectable">
      <div style="text-align:
      left; font-size: 1.3em;">
        Selecciones de modos&
      nbsp;
        <button onclick="
      pasteNodos()">ClipBoard
      Nodos</button>
      </div>
      <br>
      <div class="divdatos
      contenedornodos">
        <table class=tbdatos>
          <tbody id="resultados"
      >
          </tbody>
        </table>
      </div>
    </td>
  </tr>
</table>
<div id="selrecdiv" class="
  selrecdiv"></div>
</body>
</html>
```

7.4.2 JavaScript code

```
var _objs=new Array(); //
    Collection of objects for
    global use
var _cobjs=new Array(); //
    Collection of objects for
    copy-paste use
var _auxobjs=new Array(); //
    Objects auxiliary for tab
    selection
var sel=0;
var currentSel;
var RED="#f70000";
var BLACK="black";
var BLUE="#0785EE";
var VERDE="#1B4ADE";
//_____
$(document).ready(readyFn);
//_____
function readyFn()
{
  $('#areatrabajo').css('cursor'
    ,'crosshair');
  $('#areatrabajo').mousedown(
    function(e)
  {
    _x1 = parseInt(e.pageX);
    _y1 =parseInt(e.pageY);
    _x2=_x1;
    _y2=_y1;
    $(document).mousemove(
  function(event)
    {
      // Moves working area
    scrollings
      _x2 = event.pageX;
      _y2 = event.pageY;
      var ancho=Math.abs(_x2-
    _x1);
      var alto=Math.abs(_y2-_y1
    );
      oX=(_x2>=_x1)? _x1 : _x2;
    // Calculate X origin
    coordinate
      oY=(_y2>=_y1)? _y1 : _y2;
    // Calculate Y origin
    coordinate
      // Selecting rectangle
    coordinates
      _hx1=oX;
      _hx2=_hx1+ancho;
      _hy1=oY;
      _hy2=_hy1+alto;
      // Displays selecting
    rectangle
      $('#selrecdiv').show();
      $('#selrecdiv').height(
    alto);
      $('#selrecdiv').width(
    ancho);
      $("#selrecdiv").css('top',
    oY);
      $("#selrecdiv").css('left'
    , oX);
      desactivaSeleccion();
      $("#svgNodos .nodo").each(
    function(){
        if(seleccionable($(this)
    ,_hx1,_hy1,_hx2,_hy2)){
          marcaNodo($(this));
          _objs.push($(this));
    // Guarda elemento g
        }
      });
    });
    $(document).on("mouseup",
    function() {
      $('#selrecdiv').hide();
      $(document).off("
    mousemove");
      // Clones selecting tabs
    to generate a new z-index
      var nwnodos="";
      $(_objs).each(function(){
        nwnodos+=$(this).attr("
    id")+';'
      });
      if(nwnodos.length){
        nwnodos=nwnodos.
    substring(0, nwnodos.length-
    1);
        $('#resultados').append(
    html);
      }
      sel++;
```

Script 7.2 JavaScript Code.

This function runs when the html archive is downloaded to the navigator. What follows generates the node-selecting rectangle. The rectangle's first corner is generated in the beginning where the user first clicks, and its diagonally opposite corner is generated where the user unclicks. When selected, nodes turn blue and after unclicking, a new entrance in the table with the id of the selected nodes is created. This comparison is also done in JavaScript.

This generates a line with the html code inside the label `table´, which adds a new row with three buttons (delete, see nodes and copy to clipboard), apart from adding to the same line a new column with the nodes id.

```
        var html='<tr id="tr-'+sel+'"><th><button onclick="eliminaseleccion('+sel+')">Eliminar</button></th>';
        html+='<th ><button onclick="vernodos('+sel+')">Ver nodos</button></th>';
        html+='<th ><button onclick="clipboradLinea (\''+nwnodos+'\')">ClipBoard</button></th>';
        html+='<td id="td-'+sel+'"><span id="span-'+sel+'">'+nwnodos+'</span></td></tr>';
        $('#resultados').append(html);
        currentSel=document.getElementById("td-"+sel);
        $(currentSel).css("background-color","red");
        $(currentSel).css("color","white");
        $(currentSel).css("font-weight","600");
        $(document).off("mouseup");
      });
    });
  }
  //_____________________________
  //
  // Deactivates the selection of tablatures and deletes selection buffer
  //_____________________________
  function desactivaSeleccion()
  {
    // Deactivates tablatures selection
    $(_objs).each(function(){
      desmarcaNodo($(this));
    });
    _objs.splice(0);

    for(var i=0;i<_cobjs.length;i++){
      $("#svgNodos #"+_cobjs[i]).attr("fill","RED");
    }
    if(currentSel){
      $(currentSel).css("background-color","white");
      $(currentSel).css("color","black");
      $(currentSel).css("font-weight","400");
    }
  }
  //_____________________________
  function marcaNodo(nodo)
  {
    $(nodo).attr("fill","VERDE");
  }
  //_____________________________
  function desmarcaNodo(nodo)
  {
    $(nodo).attr("fill","RED");
  }
  //_____________________________
  function eliminaseleccion(sel)
  {
    if(!confirm("ATENCIÓN: Va a eliminar esta seleccion ¿Está seguro?")) return
    $(document).off("mouseup");
    $("#tr-"+sel).remove();
    for(var i=0;i<_cobjs.length;i++){
      $("#svgNodos #"+_cobjs[i]).attr("fill","RED");
    }
    desactivaSeleccion();
  }
  //_____________________________
  function vernodos(sel)
  {
    desactivaSeleccion();
    currentSel=document.getElementById("td-"+sel);
    $(currentSel).css("background-color","red");
    $(currentSel).css("color","white");
    $(currentSel).css("font-weight","600");
    var nodosel=$("#span-"+sel).html();
    _cobjs=new Array(); // Colección objetos para uso de copy paste
    _cobjs = nodosel.split(';');
    for(var i=0;i<_cobjs.length;i++){
      $("#svgNodos #"+_cobjs[i]).attr("fill","VERDE");
    }
  }
  //_____________________________
  // Proves whether a tab is inside the selected region
  //_____________________________
  function seleccionable(obj,cx1,cy1,cx2,cy2)
  {
    if($(obj).offset().top>cy2 || $(obj).offset().top<cy1)
      return;
    var x=$(obj).offset().left
    var w=parseFloat($(obj).attr("r"))*2;
    var y=$(obj).offset().top
    var h=parseFloat($(obj).attr("r"))*2;
    if(cx1<=x && (parseFloat(x)+parseFloat(w))<=cx2){
      if(cy1<=y && (parseFloat(y)+parseFloat(h))<=cy2)
        return(true);
    }
    return(false);
  }
  //_____________________________
  // Copies nodes to the clipboard
  //_____________________________
  function pasteNodos()
  {
    var nodos="";
    $('#resultados span').each(
      function(){
      nodos+=$(this).html()+";";
    });
    copyTextToClipboard(nodos);
    alert("Se han copiado los nodos en el portapapeles");
  }
  //_____________________________
  function clipboradLinea(cadena)
  {
    if(copyTextToClipboard(cadena))
      alert("Se han copiado nodos de esta selecci ón al portapapeles");
  }
  //_____________________________
  function copyTextToClipboard(text) {
    if (!navigator.clipboard) {
    alert("ATENCIÓN.- Este navegador no soporta copiar texto en el portapapeles. Utilice otro, por ejemplo Google Chrome");
    return(false);
    fallbackCopyTextToClipboard(text);
    return;
    }
    navigator.clipboard.writeText(text).then(function() {
    console.log('Async: Copying to clipboard was successful!');
    }, function(err) {
    console.error('Async: Could not copy text: ', err);
    });
    return(true);
  }
```

8 Appendix: Tables

8.1 Score Table: a cases Stability (normal criterion 2)

Table 8.1 Scores for a cases Stability (normal criterion 2).

		1st Crown				
	4E Stab.	***1***	***2***	***3***	***4***	***5***
2nd Crown	***0***	0.662867074	0.664792432	0.664880699	0.662006348	0.663763655
	1	0.65999112	0.659992632	0.660796979	0.647140086	0.651216154
	2	0.659992598	0.65999112	0.660796979	0.647140086	0.650888779
	3	0.654986171	0.654986171	0.655823419	0.642888608	0.646191793
	4	0.650442822	0.650442822	0.650974594	0.654002548	0.65997449
	5	0.674061899	0.674061899	0.654668442	0.653515753	0.632372149

		1st Crown				
	6E Stab.	***1***	***2***	***3***	***4***	***5***
2nd Crown	***0***	0.663900487	0.663900487	0.664808487	0.663437268	0.663609073
	1	0.660881107	0.660881107	0.661482688	0.651786213	0.653897431
	2	0.660881107	0.660881107	0.661482689	0.651786213	0.760531621
	3	0.656853214	0.656853214	0.66355396	0.647718998	0.649467479
	4	0.656413599	0.656413599	0.65675609	0.656288111	0.661353971
	5	0.667297488	0.667297488	0.668075248	0.656489927	0.649445379

		1st Crown				
	8E Stab.	***1***	***2***	***3***	***4***	***5***
2nd Crown	***0***	0.662714516	0.662714516	0.663577814	0.66264669	0.663753458
	1	0.659855144	0.66138088	0.660419219	0.65361573	0.652080644
	2	0.659855143	0.66138088	0.660419219	0.65361573	0.652052295
	3	0.655680035	0.657188036	0.657458109	0.649702298	0.649467703
	4	0.657073172	0.745375019	0.657364431	0.656573614	0.660532524
	5	0.667160605	0.667160605	0.667844502	0.656356003	0.656740674

		1st Crown				
	12E Stab.	***1***	***2***	***3***	***4***	***5***
	0	0.663731231	0.663731231	0.664507975	0.663651941	0.64748072
	1	0.661010367	0.663773113	0.6615218	0.655266717	0.657883186
2nd Crown	***2***	0.661010367	0.663773113	0.6615218	0.655266717	0.657873389
	3	0.65727861	0.660009566	0.657824262	0.651583365	0.654220569
	4	0.657273219	0.729622618	0.65750124	0.656943804	0.656345757
	5	0.661164036	0.663845391	0.661450194	0.656866707	0.655981296

		1st Crown				
	24E Stab.	***1***	***2***	***3***	***4***	***5***
	0	0.670563026	0.663725995	0.66448514	0.663651941	0.646842933
	1	0.660714185	0.66341278	0.661182001	0.655266717	0.65809449
2nd Crown	***2***	0.660714185	0.66341278	0.661182001	0.655266717	0.658093391
	3	0.657220132	0.659887725	0.657724405	0.651583365	0.654458495
	4	0.670190027	0.730816512	0.65700697	0.656943804	0.656383712
	5	0.660992139	0.660992139	0.661663657	0.656866707	0.656177615

8.2 Score Table: a cases Convenience (normal criterion 2)

Table 8.2 Scores for a cases Convenience criterion (normal criterion 2).

		1st Crown				
	4E Par. Conv.	***1***	***2***	***3***	***4***	***5***
	0	0.818455561	0.807222396	0.806359551	0.805642849	0.806523585
	1	0.747594263	0.749362988	0.748679328	0.716603627	0.718064673
2nd Crown	***2***	0.749362459	0.749362089	0.748679328	0.716603627	0.717982834
	3	0.747226678	0.747226678	0.746551762	0.713772378	0.715482275
	4	0.773869079	0.773869079	0.77311785	0.772106039	0.748031671
	5	0.69688269	0.69688269	0.746705577	0.77331063	0.741125443

		1st Crown				
	6E Par. Conv.	***1***	***2***	***3***	***4***	***5***
	0	0.805672047	0.805672047	0.804573127	0.804232081	0.748719416
	1	0.775594144	0.775594144	0.774418676	0.742889835	0.772079789
2nd Crown	***2***	0.775594537	0.775594537	0.774418676	0.742889835	0.742210059
	3	0.773261304	0.773260901	0.773610233	0.739220478	0.769203924
	4	0.800928955	0.800928955	0.799688303	0.796918959	0.773944384
	5	0.77675642	0.77675642	0.775624608	0.799179842	0.77096719

		1st Crown				
	8E Par. Conv.	1	2	3	4	5
2nd Crown	0	0.804051489	0.804051489	0.802499006	0.802266414	0.802541549
	1	0.745791341	0.773766583	0.744163956	0.768473117	0.687408497
	2	0.745791341	0.773766583	0.744163956	0.768473117	0.715179183
	3	0.742979209	0.770949942	0.741655322	0.763957496	0.684544826
	4	0.798883357	0.819977597	0.797187852	0.79345364	0.743750804
	5	0.74717619	0.74717619	0.745578814	0.794504445	0.77146372

		1st Crown				
	12E Par. Conv.	1	2	3	4	5
2nd Crown	0	0.801653383	0.801653383	0.795216277	0.796328469	0.792284657
	1	0.770321795	0.798526437	0.767797133	0.788705796	0.736015082
	2	0.770321795	0.798526437	0.767797133	0.788705796	0.736012633
	3	0.766736337	0.794932846	0.76422023	0.782479982	0.731120648
	4	0.791859397	0.809056092	0.786611432	0.782267272	0.761198098
	5	0.769918093	0.798120299	0.739559349	0.785127259	0.768622456

		1st Crown				
	24E Par. Conv.	1	2	3	4	5
2nd Crown	0	0.739788738	0.79369446	0.778853309	0.796328469	0.778421109
	1	0.704082087	0.760052784	0.698893999	0.788705796	0.758098583
	2	0.704082087	0.760052784	0.698893999	0.788705796	0.758098308
	3	0.697903534	0.753866351	0.692724561	0.782479982	0.739948204
	4	0.78315312	0.797433689	0.763942171	0.782267272	0.750597525
	5	0.731265742	0.731265742	0.726128598	0.785127259	0.758061471

8.3 Score Table: b cases Stability (normal criterion 2)

Table 8.3 Scores for b cases Convenience criterion (normal criterion 2).

		1st Crown				
	4E Stab.	1	2	3	4	5
2nd Crown	0	0.662867074	0.664792432	0.664880699	0.661266908	0.665107457
	1	0.674264335	0.674264335	0.675507538	0.64624417	0.639912003
	2	0.674264335	0.674264335	0.675507538	0.646244169	0.639716321
	3	0.672341151	0.672341151	0.673588843	0.642046482	0.636444725
	4	0.650125896	0.650125896	0.650679832	0.652028704	0.659195358
	5	0.674559229	0.674559229	0.654362506	0.653515753	0.659813078

		1st Crown				
	6E Stab.	***1***	***2***	***3***	***4***	***5***
	0	0.665375549	0.665375549	0.666223756	0.661735864	0.662289038
	1	0.666021054	0.666021054	0.6668002	0.650739862	0.643650612
2nd Crown	**2**	0.666021054	0.666021054	0.6668002	0.650739862	0.580380612
	3	0.662405569	0.662405569	0.663211672	0.646710425	0.640608352
	4	0.65575344	0.65575344	0.656218504	0.655297786	0.66031053
	5	0.666922595	0.666922595	0.667750216	0.655591078	0.660759765

		1st Crown				
	8E Stab.	***1***	***2***	***3***	***4***	***5***
	0	0.662004178	0.662004178	0.662831016	0.661149374	0.662213948
	1	0.659257601	0.659257601	0.659857471	0.652002474	0.65167463
2nd Crown	**2**	0.659257601	0.659257601	0.659857471	0.652002474	0.577445738
	3	0.655152473	0.655152473	0.655770069	0.648143501	0.64909434
	4	0.65500209	0.655266321	0.655410151	0.655040801	0.659215701
	5	0.660829554	0.660829554	0.661484663	0.655330658	0.659829776

		1st Crown				
	12E Stab.	***1***	***2***	***3***	***4***	***5***
	0	0.661888992	0.661888993	0.662711356	0.661291459	0.66143681
	1	0.660296948	0.660296948	0.660850756	0.65379232	0.655626391
2nd Crown	**2**	0.660296948	0.660296948	0.660850756	0.65379232	0.655372469
	3	0.656616962	0.656616962	0.657193442	0.650180929	0.652058427
	4	0.655629587	0.661692331	0.656036148	0.655600587	0.654003501
	5	0.660528319	0.660528319	0.660874644	0.655645975	0.654024527

		1st Crown				
	24E Stab.	***1***	***2***	***3***	***4***	***5***
	0	0.655531694	0.661351013	0.662128576	0.660539344	0.660601902
	1	0.660188144	0.660188144	0.660723744	0.65361888	0.655230592
2nd Crown	**2**	0.660188144	0.660188144	0.660723744	0.65361888	0.655229492
	3	0.656762103	0.656762103	0.657318235	0.650091295	0.651802182
	4	0.668706197	0.656033885	0.655803807	0.654867356	0.65340676
	5	0.660242377	0.660242377	0.660773776	0.655106776	0.653468682

8.4 Score Table: b cases Convenience (normal criterion 2)

Table 8.4 Scores for b cases Convenience Criterion (normal criterion 2).

		1st Crown				
	4E Par. Conv.	***1***	***2***	***3***	***4***	***5***
	0	0.755957352	0.744723522	0.743861292	0.699047915	0.705534059
	1	0.629915079	0.629915079	0.629341707	0.69263962	0.686576422
2nd Crown	**2**	0.629915079	0.629915079	0.629341707	0.692639619	0.68652752
	3	0.62855011	0.62855011	0.62797786	0.689821819	0.684383317
	4	0.726312445	0.726312445	0.725566754	0.675669243	0.692815527
	5	0.668345065	0.668345065	0.700858409	0.693945551	0.699822284

		1st Crown				
	6E Par. Conv.	***1***	***2***	***3***	***4***	***5***
	0	0.715873296	0.715873296	0.714759094	0.685148248	0.687718013
	1	0.66696701	0.66696701	0.665835497	0.67351536	0.678353413
2nd Crown	**2**	0.666967011	0.666967011	0.665835497	0.67351536	0.657804691
	3	0.66473688	0.66473688	0.663612103	0.669855447	0.675824491
	4	0.702194401	0.702194401	0.700984393	0.673682888	0.696244682
	5	0.693421462	0.693421462	0.69230211	0.684366292	0.705419745

		1st Crown				
	8E Par. Conv.	***1***	***2***	***3***	***4***	***5***
	0	0.697463886	0.697463886	0.695902248	0.675938825	0.685488809
	1	0.6906206	0.6906206	0.689002161	0.672347534	0.668113059
2nd Crown	**2**	0.6906206	0.6906206	0.689002161	0.672347534	0.672272679
	3	0.687825961	0.687825961	0.686211953	0.667845621	0.665257552
	4	0.682008468	0.682074523	0.680342159	0.648549563	0.657619947
	5	0.679133076	0.679133076	0.677528505	0.667852794	0.66882564

		1st Crown				
	12E Par. Conv.	***1***	***2***	***3***	***4***	***5***
	0	0.692124897	0.692124897	0.685678127	0.686804457	0.682702052
	1	0.701929582	0.721495855	0.699404114	0.672017178	0.656945361
2nd Crown	**2**	0.701929582	0.721495855	0.699404114	0.672017178	0.656942911
	3	0.698343752	0.71790069	0.695826816	0.665791515	0.65205058
	4	0.682459273	0.699403505	0.677210535	0.672866311	0.643135169
	5	0.690534919	0.708731186	0.687954559	0.675726604	0.66002896

		1st Crown				
	24E Par. Conv.	***1***	***2***	***3***	***4***	***5***
	0	0.669171491	0.684168966	0.669318068	0.686804457	0.668887595
	1	0.662398985	0.680584638	0.657210186	0.672017178	0.660866625
2nd Crown	**2**	0.662398985	0.680584638	0.657210186	0.672017178	0.66086635
	3	0.656219929	0.674396512	0.651040211	0.665791515	0.642715866
	4	0.673765072	0.687850213	0.654540095	0.672866311	0.642004397
	5	0.661805036	0.661805036	0.656667205	0.675726604	0.649468373

8.5 Adjustment coefficients for a cases

Table 8.5 One-crown a cases.

4E	**type 1 type 0**	**type 2 type 0**	**type 3 type 0**	**type 4 type 0**	**type 5 type 0**
p_1	0.699316396	0.721370467	0.725865491	0.693738188	0.692766707
p_0	-0.008376937	-0.016454341	-0.017262345	-0.005477516	-0.005824042
6E	**type 1 type 0**	**type 2 type 0**	**type 3 type 0**	**type 4 type 0**	**type 5 type 0**
p_1	0.714241418	0.714241418	0.71739044	0.700172979	0.700123043
p_0	-0.011332475	-0.011332475	-0.011799581	-0.005750364	-0.006089905
8E	**type 1 type 0**	**type 2 type 0**	**type 3 type 0**	**type 4 type 0**	**type 5 type 0**
p_1	0.709116282	0.709116282	0.711741058	0.702693161	0.702223561
p_0	-0.00849959	-0.00849959	-0.008855271	-0.005964584	-0.00598857
12E	**type 1 type 0**	**type 2 type 0**	**type 3 type 0**	**type 4 type 0**	**type 5 type 0**
p_1	0.711496108	0.711496108	0.713803468	0.704545669	0.703920464
p_0	-0.008575322	-0.008575322	-0.008900085	-0.006041095	-0.006002726
24E	**type 1 type 0**	**type 2 type 0**	**type 3 type 0**	**type 4 type 0**	**type 5 type 0**
p_1	0.709116282	0.709116282	0.711741058	0.702693161	0.702223561
p_0	-0.00849959	-0.00849959	-0.008855271	-0.005964584	-0.00598857

Table 8.6 Two crown a cases.

			1^{st} **crown type**				
	4E		**1**	**2**	**3**	**4**	**5**
2^{nd} **crown type**	**1**	p_1	1.6867245	1.6867244	1.6894049	1.6703374	1.6697376
		p_2	-0.0017726	-0.0017726	-0.0017083	-0.0032327	-0.0031663
		p_0	-0.0050516	-0.0050517	-0.0053385	-0.0005834	-0.0008996
	2	p_1	1.6871925	1.6867245	1.6894049	1.6703374	1.6697527
		p_2	0.0003820	-0.0017726	-0.0017083	-0.0032327	-0.0030488
		p_0	-0.0057496	-0.0050516	-0.0053385	-0.0005834	-0.0009332
	3	p_1	1.6846197	1.6846197	1.6867245	1.6682768	1.6678813
		p_2	0.0003312	0.0003312	-0.0017726	-0.0015864	-0.0014140
		p_0	-0.0054791	-0.0054791	-0.0050516	-0.0009105	-0.0012117
	4	p_1	1.6769952	1.6769952	1.6791437	1.6827517	1.6883567
		p_2	-0.0025177	-0.0025177	-0.0024742	-0.0023952	-0.0018377
		p_0	-0.0018227	-0.0018227	-0.0019846	-0.0024714	-0.0048130
	5	p_1	1.7074728	1.7074728	1.6787648	1.6826931	1.6880808
		p_2	0.0006530	0.0006530	-0.0021046	-0.0020918	-0.0013794
		p_0	-0.0126425	-0.0126425	-0.0023038	-0.0027197	-0.0052554

			1st crown type				
	6E		**1**	**2**	**3**	**4**	**5**
	1	p_1	1.6961270	1.6961271	1.6976988	1.6826332	1.6849768
		p_2	-0.0010747	-0.0010750	-0.0010529	-0.0026601	-0.0024295
		p_0	-0.0060096	-0.0060093	-0.0061239	-0.0018505	-0.0024697
	2	p_1	1.6961271	1.6961271	1.6976988	1.6826332	1.6835046
		p_2	-0.0010750	-0.0010750	-0.0010529	-0.0026601	-0.0196051
		p_0	-0.0060093	-0.0060093	-0.0061239	-0.0018505	0.0093866
2nd	**3**	p_1	1.6938452	1.6938452	1.6953310	1.6803116	1.6828005
crown		p_2	0.0006449	0.0006449	0.0006574	-0.0010668	-0.0008077
type		p_0	-0.0062827	-0.0062827	-0.0063859	-0.0021342	-0.0027296
	4	p_1	1.6901831	1.6901831	1.6917266	1.6913863	1.6968996
		p_2	-0.0022881	-0.0022880	-0.0022329	-0.0018295	-0.0014130
		p_0	-0.0031221	-0.0031221	-0.0032433	-0.0033897	-0.0056025
	5	p_1	1.7062876	1.7005141	1.7082988	1.6913545	1.6970652
		p_2	-0.0006975	-0.0218790	-0.0006694	-0.0017554	-0.0009870
		p_0	-0.0087947	0.0075208	-0.0090019	-0.0034818	-0.0060645

			1st crown type				
	8E		**1**	**2**	**3**	**4**	**5**
	1	p_1	1.6960794	1.6960793	1.6979643	1.6882044	1.6966203
		p_2	-0.0018152	-0.0018103	-0.0018200	-0.0027666	-0.0019847
		p_0	-0.0050894	-0.0050928	-0.0052551	-0.0022779	-0.0050105
	2	p_1	1.6960794	1.6960793	1.6979643	1.6882044	1.6965937
		p_2	-0.0018152	-0.0018103	-0.0018200	-0.0027666	-0.0019462
		p_0	-0.0050894	-0.0050928	-0.0052551	-0.0022779	-0.0050268
2nd	**3**	p_1	1.6936084	1.6936083	1.6953341	1.6858175	1.6940070
crown		p_2	-0.0001006	-0.0000957	-0.0000581	-0.0011423	-0.0002780
type		p_0	-0.0053612	-0.0053646	-0.0055433	-0.0025743	-0.0052755
	4	p_1	1.6924175	1.6885851	1.6943677	1.6932596	1.6986479
		p_2	-0.0019520	-0.0193744	-0.0018183	-0.0017201	-0.0014134
		p_0	-0.0033424	0.0089982	-0.0035742	-0.0034970	-0.0055818
	5	p_1	1.6981914	1.6981914	1.6996977	1.6932205	1.6985796
		p_2	-0.0011186	-0.0011186	-0.0010269	-0.0016343	-0.0013586
		p_0	-0.0057929	-0.0057929	-0.0059644	-0.0035734	-0.0056695

			1st crown type				
	12E		**1**	**2**	**3**	**4**	**5**
	1	p_1	1.6999712	1.6988288	1.7015616	1.6913745	1.7188120
		p_2	-0.0013853	-0.0124948	-0.0013231	-0.0022309	-0.0014770
		p_0	-0.0056876	0.0021801	-0.0058687	-0.0028155	-0.0109862
	2	p_1	1.6999712	1.6988288	1.7015616	1.6913745	1.7188125
		p_2	-0.0013853	-0.0124948	-0.0013231	-0.0022309	-0.0014512
		p_0	-0.0056876	0.0021801	-0.0058687	-0.0028155	-0.0110062
2^{nd}	**3**	p_1	1.6974904	1.6963475	1.6990279	1.6891046	1.7167820
crown		p_2	0.0002027	-0.0109115	0.0002470	-0.0007732	-0.0001593
type		p_0	-0.0059221	0.0019489	-0.0060898	-0.0030457	-0.0111703
	4	p_1	1.6940532	1.6981477	1.6955171	1.6947369	1.7188618
		p_2	-0.0017718	-0.0242463	-0.0017068	-0.0018564	-0.0015532
		p_0	-0.0034616	0.0102150	-0.0036098	-0.0034586	-0.0108669
	5	p_1	1.7015837	1.7008330	1.7020216	1.6947319	1.7186351
		p_2	-0.0011679	-0.0125103	-0.0011583	-0.0018965	-0.0012491
		p_0	-0.0060627	0.0017775	-0.0060170	-0.0034501	-0.0110586

			1st crown type				
	24E		**1**	**2**	**3**	**4**	**5**
	1	p_1	1.7009137	1.6995978	1.7023247	1.6919146	1.7235422
		p_2	-0.0012056	-0.0123571	-0.0011483	-0.0021321	-0.0012502
		p_0	-0.0057354	0.0020330	-0.0058955	-0.0028287	-0.0120399
	2	p_1	1.7009137	1.6995978	1.7023247	1.6919146	1.7235411
		p_2	-0.0012056	-0.0123571	-0.0011483	-0.0021321	-0.0012496
		p_0	-0.0057354	0.0020330	-0.0058955	-0.0028287	-0.0120399
2^{nd}	**3**	p_1	1.6985027	1.6971862	1.6998871	1.6896712	1.7219162
crown		p_2	0.0002696	-0.0108863	0.0003058	-0.0007131	-0.0002321
type		p_0	-0.0059396	0.0018318	-0.0060865	-0.0030453	-0.0121463
	4	p_1	1.7146987	1.6881416	1.6967477	1.6955540	1.7198921
		p_2	-0.0011859	-0.0124858	-0.0017459	-0.0018422	-0.0015068
		p_0	-0.0092715	0.0046742	-0.0036277	-0.0034732	-0.0109356
	5	p_1	1.7010286	1.7010286	1.7025776	1.6955513	1.7198391
		p_2	-0.0013728	-0.0013728	-0.0014492	-0.0018151	-0.0014480
		p_0	-0.0056455	-0.0056455	-0.0057185	-0.0035038	-0.0109780

Table 8.7 One-crown b cases.

4E	**type 1 type 0**	**type 2 type 0**	**type 3 type 0**	**type 4 type 0**	**type 5 type 0**
p_1	0.683569908	0.693775343	0.693285007	0.754764074	0.693582651
p_0	-0.014835322	-0.006490713	-0.005586549	-0.02821319	-0.005747557
6E	**type 1 type 0**	**type 2 type 0**	**type 3 type 0**	**type 4 type 0**	**type 5 type 0**
p_1	0.699419006	0.699419006	0.701992941	0.747145291	0.69985529
p_0	-0.00590665	-0.00590665	-0.006206542	-0.022657676	-0.005526055
8E	**type 1 type 0**	**type 2 type 0**	**type 3 type 0**	**type 4 type 0**	**type 5 type 0**
p_1	0.701825963	0.701825963	0.704007907	0.774798469	0.70255688
p_0	-0.005596863	-0.005596863	-0.005826572	-0.032972224	-0.005686963
12E	**type 1 type 0**	**type 2 type 0**	**type 3 type 0**	**type 4 type 0**	**type 5 type 0**
p_1	0.7041073	0.7041073	0.706035086	0.756101823	0.755717653
p_0	-0.005666085	-0.005666085	-0.005878402	-0.024650382	-0.024717687
24E	**type 1 type 0**	**type 2 type 0**	**type 3 type 0**	**type 4 type 0**	**type 5 type 0**
p_1	0.697316181	0.705272161	0.706944907	0.760888331	0.756241747
p_0	-0.002660257	-0.00558706	-0.00576177	-0.026228621	-0.024480946

8.6 Adjustment coefficients for b cases

Table 8.8 Two-crown b cases.

			1^{st} **crown type**				
	4E		**1**	**2**	**3**	**4**	**5**
	1	p_1	1.7044291	1.7044291	1.7078642	1.6705268	1.6697719
		p_2	-0.0031716	-0.0031716	-0.0030514	-0.0033376	-0.0028602
		p_0	-0.0093358	-0.0093358	-0.0098965	-0.0001454	-0.0009216
	2	p_1	1.7044291	1.7044291	1.7078642	1.6705268	1.6697881
		p_2	-0.0031716	-0.0031716	-0.0030514	-0.0033376	-0.0027438
		p_0	-0.0093358	-0.0093358	-0.0098965	-0.0001454	-0.0009540
2^{nd}	**3**	p_1	1.7019359	1.7019359	1.7052277	1.6684431	1.6678812
crown		p_2	-0.0005714	-0.0005714	-0.0005031	-0.0017006	-0.0011116
type		p_0	-0.0100269	-0.0100269	-0.0105351	-0.0004649	-0.0012308
	4	p_1	1.6772839	1.6772839	1.6792787	1.6829170	1.6885479
		p_2	-0.0025825	-0.0025825	-0.0025204	-0.0024255	-0.0019403
		p_0	-0.0014264	-0.0014264	-0.0015817	-0.0020208	-0.0043773
	5	p_1	1.7076193	1.7076193	1.6788509	1.6826931	1.6880808
		p_2	0.0002524	0.0002524	-0.0020115	-0.0020918	-0.0013794
		p_0	-0.0121905	-0.0121905	-0.0021855	-0.0027197	-0.0052554

			1st crown type				
	4E		**1**	**2**	**3**	**4**	**5**
	1	p_1	1.6963644	1.6963644	1.6979081	1.6827237	1.6851085
		p_2	-0.0014079	-0.0014079	-0.0013688	-0.0027481	-0.0027112
		p_0	-0.0056197	-0.0056197	-0.0057414	-0.0013880	-0.0019108
	2	p_1	1.6963644	1.6963644	1.6979081	1.6827237	1.4226718
		p_2	-0.0014079	-0.0014079	-0.0013688	-0.0027481	0.1696187
		p_0	-0.0056197	-0.0056197	-0.0057414	-0.0013880	-0.0458324
2nd crown type	**3**	p_1	1.6934941	1.6934941	1.6955364	1.6803832	1.6829188
		p_2	0.0003033	0.0003033	0.0003573	-0.0011741	-0.0011093
		p_0	-0.0057921	-0.0057921	-0.0060162	-0.0016572	-0.0021583
	4	p_1	1.6903826	1.6903826	1.6919180	1.6912877	1.6973923
		p_2	-0.0023728	-0.0023728	-0.0023251	-0.0020264	-0.0012877
		p_0	-0.0026747	-0.0026747	-0.0027890	-0.0027916	-0.0053259
	5	p_1	1.7065202	1.7065202	1.7084621	1.6912537	1.6966968
		p_2	-0.0009088	-0.0009088	-0.0008964	-0.0020129	-0.0012330
		p_0	-0.0083248	-0.0083248	-0.0085068	-0.0028256	-0.0056375

			1st crown type				
	8E		**1**	**2**	**3**	**4**	**5**
	1	p_1	1.6964109	1.6964109	1.6981849	1.6884181	1.6972325
		p_2	-0.0019570	-0.0019570	-0.0019055	-0.0028454	-0.0022469
		p_0	-0.0046484	-0.0046484	-0.0048355	-0.0018030	-0.0045800
	2	p_1	1.6964109	1.6964109	1.6981849	1.6884181	1.2072862
		p_2	-0.0019570	-0.0019570	-0.0019055	-0.0028454	0.3958576
		p_0	-0.0046484	-0.0046484	-0.0048355	-0.0018030	-0.1320626
2nd crown type	**3**	p_1	1.6938288	1.6938288	1.6955355	1.6860232	1.6946064
		p_2	-0.0002127	-0.0002127	-0.0001703	-0.0012595	-0.0005486
		p_0	-0.0049281	-0.0049281	-0.0051060	-0.0020750	-0.0048363
	4	p_1	1.6929775	1.6932707	1.6945762	1.6934519	1.6988877
		p_2	-0.0019315	-0.0021166	-0.0018921	-0.0019374	-0.0015389
		p_0	-0.0029760	-0.0029289	-0.0031070	-0.0029112	-0.0050890
	5	p_1	1.6983308	1.6983308	1.6998433	1.6934486	1.6987667
		p_2	-0.0015662	-0.0015662	-0.0014814	-0.0018590	-0.0014089
		p_0	-0.0051553	-0.0051553	-0.0053219	-0.0029589	-0.0052688

			1^{st} crown type				
	12E		**1**	**2**	**3**	**4**	**5**
	1	p_1	1.7002062	1.7002062	1.7017843	1.6915862	1.7229599
		p_2	-0.0015043	-0.0015043	-0.0014491	-0.0024411	-0.0011820
		p_0	-0.0052268	-0.0052268	-0.0054004	-0.0022524	-0.0117360
	2	p_1	1.7002062	1.7002062	1.7017843	1.6915862	1.7229753
		p_2	-0.0015043	-0.0015043	-0.0014491	-0.0024411	-0.0010528
		p_0	-0.0052268	-0.0052268	-0.0054004	-0.0022524	-0.0118722
2^{nd}	**3**	p_1	1.6977064	1.6977064	1.6992415	1.6893135	1.7211758
crown		p_2	0.0000539	0.0000539	0.0000972	-0.0010180	-0.0000817
type		p_0	-0.0054401	-0.0054401	-0.0056053	-0.0024580	-0.0118277
	4	p_1	1.6942042	1.6984651	1.6956664	1.6949233	1.7191042
		p_2	-0.0019258	-0.0040722	-0.0018820	-0.0020568	-0.0014229
		p_0	-0.0029277	-0.0026330	-0.0030609	-0.0028376	-0.0105581
	5	p_1	1.7012525	1.7012525	1.7015740	1.6948632	1.7189111
		p_2	-0.0014142	-0.0014142	-0.0014253	-0.0020280	-0.0012119
		p_0	-0.0053665	-0.0053665	-0.0052868	-0.0029114	-0.0107252

			1^{st} crown type				
	24E		**1**	**2**	**3**	**4**	**5**
	1	p_1	1.7011286	1.7011286	1.7025396	1.6920406	1.7238782
		p_2	-0.0014361	-0.0014361	-0.0013915	-0.0023758	-0.0010600
		p_0	-0.0051682	-0.0051682	-0.0053190	-0.0021816	-0.0117807
	2	p_1	1.7011286	1.7011286	1.7025396	1.6920406	1.7238772
		p_2	-0.0014361	-0.0014361	-0.0013915	-0.0023758	-0.0010594
		p_0	-0.0051682	-0.0051682	-0.0053190	-0.0021816	-0.0117807
2^{nd}	**3**	p_1	1.6987146	1.6987146	1.7001009	1.6897962	1.7222427
crown		p_2	-0.0000084	-0.0000084	0.0000263	-0.0009898	-0.0000823
type		p_0	-0.0053391	-0.0053391	-0.0054850	-0.0023739	-0.0118554
	4	p_1	1.7150194	1.6976090	1.6969488	1.6956974	1.7201834
		p_2	-0.0010933	-0.0032487	-0.0020059	-0.0020049	-0.0014113
		p_0	-0.0088894	-0.0025965	-0.0029757	-0.0029124	-0.0105513
	5	p_1	1.7012261	1.7012261	1.7026206	1.6957490	1.7201025
		p_2	-0.0015637	-0.0015637	-0.0015137	-0.0020238	-0.0013169
		p_0	-0.0050920	-0.0050920	-0.0052433	-0.0028832	-0.0106734

8.7 Some images of outliers

These are a few images of outliers that end up appearing because of the sweep. As proven, no poorly shaped case enters our final analysis. Criteria bound to spot them proves to be good enough.

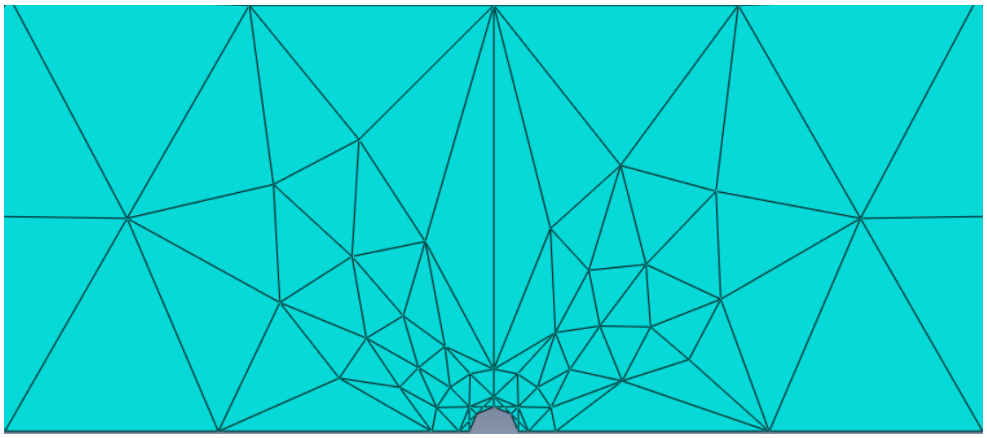


Figure 8.1 Too small α_1.

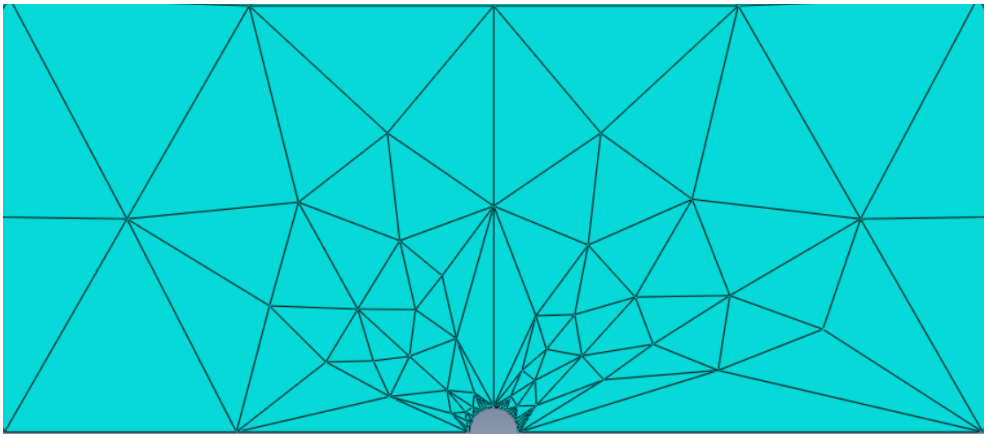


Figure 8.2 α_2 and α_3 too close.

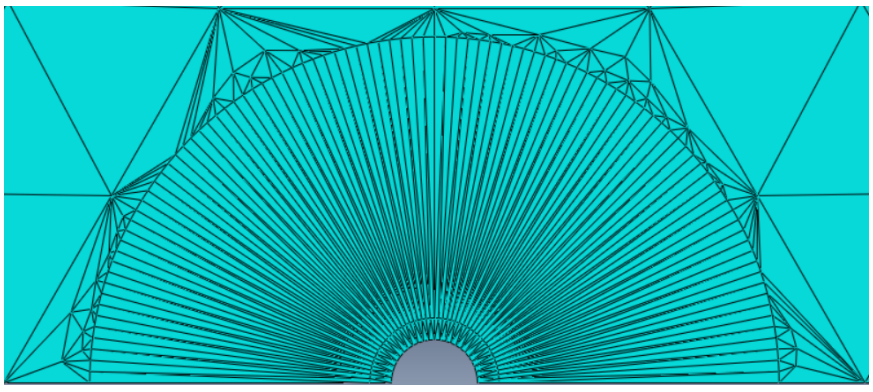


Figure 8.3 α_3 too high. The same applies to α_2.

Most outliers fall in one of these categories.

9 Appendix: Codes

Some of them are explained in Spanish.

obtain data

This code reads the .mat results files of the configurations we want to compare and extract the desired variable to plot or table them.

```
clear all, close all, clc
% clc
%% Descripcion
%resuelve la placa rectangular inicial con grieta
id_indice_real = strvcat('24E_tipo_1_tipo_1',...
'24E_tipo_2_tipo_1',...
'24E_tipo_3_tipo_1',...
'24E_tipo_4_tipo_1',...
'24E_tipo_5_tipo_1');
taula_criterio_2_index =[];
criterio_1_mejorado_index =[];
ajuste_index =[]; ajuste_par_index =[];

%% Makes lists
for puntero_real=1:size(id_indice_real,1)
%% This code finds the crown type data by the naming code
id= id_indice_real(puntero_real,:);
separadores= strfind(id,'_');
nofelements=str2num(id(1:separadores(1)-2));
if str2num(id(separadores(end)+1:end))~=0
    nofcrowns=2;
else
    nofcrowns=1;
end
tipo1=str2num(id(separadores(2)+1:separadores(3)-1));
tipo2=str2num(id(separadores(4)+1:end));
cd(['C:\Users\Usuario\Desktop\TFG (Avances y Bibliografia)\Macros\',id(1:separadores(1)-1),'\',id]);

%% This code obtains the desired results
load(id_indice_real(puntero_real,:));
% criterio_index =[criterio_index; round((max(alpha1_index)/0.05)*criterio_1)];
criterio_1_mejorado_index =[criterio_1_mejorado_index; round((max(alpha1_index)/0.05))];
id_indice_real(puntero_real,:);
taula_criterio_2_index =[taula_criterio_2_index; criterio_2_mejorado];

cd(['C:\Users\Usuario\Desktop\TFG (Avances y Bibliografia)\Macros\Codigo Matlab']);
[ajuste,ajuste_2,slope_index,max_2nd_crown_change,casos_buenos,nota_a,nota_b,criterio_1,criterio_2,criterio_3a,
    criterio_3b,criterio_4,criterio_1_num,criterio_1_den,criterio_5,Coeff_vectora,Coeff_vectorb,mean_slopes_index
    ,criterio_2_mejorado] = step_13_numerical_regressions(alpha1_index,alpha2_index,alpha3_index,
    desplazamientos_calculo,coordenadas_calculo,tipo1,tipo2,id,separadores,coordenadas,2.25596,t);
% ajuste_index=[ajuste_index; Coeff_vectora.p1 Coeff_vectora.p2];
ajuste_index=[ajuste_index; Coeff_vectora.p10 Coeff_vectora.p01 Coeff_vectora.p00];
end
```

Script 9.1 Code to obtain data.

Definition of cut vectors

This code generates the Python script lines needed to define a crown cut automatically introducing the cut type, the crown involved (inner or outer), and if the lines are sketch on s1 or s.

```
%% Este programa crea los comandos para el corte de una corona

%% Inicializar variables
clear; clc;
code = [];
xvect = [];
yvect = [];
xvect = [];

%% Parámetros por introducir
mark = 's'; % Cambiar a 's1' dependiendo del sketch
nofelements = 6; %% Número de elementos de la circunferencia inferior
xi = 'x1'; % Elegir cuál es la circunferencia interior (x1 si es primera corona, x2 si es segunda corona)
xj = 'x2'; % Elegir cuál es la circunferencia exterior (x2 si es primera corona, x3 si es segunda corona)
x_ct = 0; % coordenada x del borde de grieta
y_ct = 0; % coordenada y del borde de grieta
tipo = 4; % Elegir el tipo de la corona

%% Creación de matriz de la forma [ x_inicial1 x_final1 ; x_inicial2 x_final2 ; ...] con un segmento en cada fila
if tipo == 1,
    for s1=0:nofelements/2
        s=num2str(s1);
        sM1=num2str(s1+1);
        sm1=num2str(s1-1);
        Ds=num2str(2*s1);
        line=[xj,'[',s,'] ',xi,'[',sM1,']'];
        line2=[xi,'[',sM1,'] ',xj,'[',sM1,']'];
        xvect=strvcat(xvect,line,line2);
    end
    xvect(end-1:end,:)=[];
    for s1=nofelements/2:nofelements
        s=num2str(s1);
        sM1=num2str(s1+1);
        sm1=num2str(s1-1);
        Ds=num2str(2*s1);
        line=[xi,'[',s,'] ',xj,'[',sM1,']'];
        line2=[xj,'[',sM1,'] ',xi,'[',sM1,']'];
        xvect=strvcat(xvect,line,line2);
    end
    xvect(end-2:end,:) = [];
    xcorte = xvect;
end

if tipo == 2,
    for s1=0:nofelements/2
        s=num2str(s1);
        sM1=num2str(s1+1);
        sm1=num2str(s1-1);
        Ds=num2str(2*s1);
        line=[xi,'[',s,'] ',xj,'[',sM1,']'];
        line2=[xi,'[',sM1,'] ',xj,'[',sM1,']'];
        xvect=strvcat(xvect,line,line2);
    end
    xvect(end-1:end,:)=[];
    for s1=nofelements/2:nofelements
        s=num2str(s1);
        sM1=num2str(s1+1);
        sm1=num2str(s1-1);
        Ds=num2str(2*s1);
        line=[xj,'[',s,'] ',xi,'[',sM1,']'];
        line2=[xj,'[',sM1,'] ',xi,'[',sM1,']'];
```

```
            xvect= strvcat (xvect, line, line2);
        end
        xvect(end-2:end,:) =[];
        xcorte=xvect;
end

if tipo == 3,
        disp('add lines obtained with Type 1 plus the ones obtained with type 2')
end

if tipo == 4,
        for s1=0:nofelements-1
            s=num2str(s1);
            sM1=num2str(s1+1);
            sm1=num2str(s1-1);
            Ds=num2str(2*s1);
            Ds_M1=num2str(2*s1+1);
            DsM1=num2str(2*(s1+1));
            line=[xi,'[',s,']',xj,'[',Ds_M1,']'];
            line2=[xj,'[',Ds_M1,']',xi,'[',sM1,']'];
            line3=[xi,'[',sM1,']',xj,'[',DsM1,']'];
            xvect= strvcat (xvect, line, line2, line3);
        end
        xvect(end,:) =[];
        xcorte=xvect;
end

if tipo == 5,
        for s1=0:nofelements-1
            s=num2str(s1);
            sM1=num2str(s1+1);
            sm1=num2str(s1-1);
            Ds=num2str(2*s1);
            Ds_M1=num2str(2*s1+1);
            DsM1=num2str(2*(s1+1));
            line=[xi,'[',s,']',xj,'[',Ds_M1,']'];
            line2=[xj,'[',Ds_M1,']',xi,'[',sM1,']'];
            xvect= strvcat (xvect, line, line2);
        end
        xcorte=xvect;
end

%% Encontrar separadores
for s1 = [1: size(xvect,1)]
        separador(s1,:) = strfind(xvect(s1,:),' ');
end

%% Creación de una matriz de la forma [ y_inicial1 y_final1; y_inicial2 y_final2; ...] con un segmento en cada fila
for s1 = [1: size(xvect,1)]
        yvect(s1,:) = strrep(xvect(s1,:),'x','y');
end

%% Creación de las líneas de código
for s1 = [1: size(xvect,1)]
        codeline = [mark,'.Line(point1=(',num2str(x_ct),'+',(xvect(s1,1: separador(s1,1)-1)),',',num2str(y_ct),'+',(
          yvect(s1,1: separador(s1,1)-1)),'), point2=(',num2str(x_ct),'+',(xvect(s1, separador(s1,1)+1:end)),',',num2str(
          y_ct),'+',yvect(s1, separador(s1,1)+1:end),'))'];
        code=[code;codeline];
end

%% Mostrar
code
```

Main step

```
function [nofelements, nofcrowns, tipo1, tipo2, separadores] = step_02_obtain_parameters(id)
%% This code finds the crown type data by the naming code
separadores= strfind(id,'_');
nofelements=str2num(id(1: separadores(1)-2));
if str2num(id(separadores(end)+1:end))~=0
```

```
    nofcrowns=2;
else
    nofcrowns=1;
end
tipo1=str2num(id(separadores(2)+1:separadores(3)-1));
tipo2=str2num(id(separadores(4)+1:end));
```

Script 9.2 Code to define cut vectors.

9.1.1 step_03_name_archives

```
clear all, close all,
% clc
%% Descripcion
%resuelve la placa rectangular inicial con grieta
id_real_index = strvcat('6E_tipo_2_tipo_5');
nota_index =[];
t_index =[];
criterio_1_index =[];
criterio_2_index =[];
criterio_3a_index =[];
criterio_3b_index =[];
criterio_4_index =[];

%%

for puntero=1:size(id_real_index,1)
    clearvars -except t_index puntero id_real_index nota_index criterio_1_index criterio_2_index criterio_3a_index
      criterio_3b_index criterio_4_index ; close all;
%% Identificación de la carpeta que analizar
id=id_real_index(puntero,:); %% Nombre de la carpeta donde están los .inp
id

%% Datos a introducir
hex_sidelength=0.1;              % Introducir lado del hexágono
typeelement=1;                   % 1: con elementos singulares // 0: sin elementos singulares
vct=[0.00 0.00 0.00];            % vector con las coordenadas del vértice de la grieta
G=1; k_robin=1; NUM=400;
rcts=2.25596;                    % Right crack tip solution

%% Inicializar variables
[name_index, file_index, alpha1_index, alpha2_index, alpha3_index, desplazamientos_calculo, coordenadas_calculo,
      singular_FE_sidelength_index, new_entrance_inp_index] = step_01_set_variables();

%% Obtener parámetros del nombre
[nofelements, nofcrowns, tipo1, tipo2, separadores] = step_02_obtain_parameters(id);

% cd(['C:\Users\Usuario\Desktop\TFG (Avances y Bibliografia)\Macros\', id(1:separadores(1)-1),'\', id]);
% load([id,'.mat']);

%% Índice de nombres
[alpha1_index, alpha2_index, alpha3_index, singular_FE_sidelength_index, file_index, name_index, ncasos] =
      step_03_name_archives(nofelements, nofcrowns, tipo1, tipo2, hex_sidelength, id, name_index, file_index,
      singular_FE_sidelength_index, alpha1_index, alpha2_index, alpha3_index, separadores);

% Bucle para análisis de todos los casos de un estudio
for s=1:size(file_index,1)
close all;

%% Cambio de directorio de trabajo (cambiar al del ordenador que lo utilice)
cd(['C:\Users\Usuario\Desktop\TFG (Avances y Bibliografia)\Macros\', id(1:separadores(1)-1), '\', id]);

%% Particularizamos los valores de ciertos parámetros al caso del índice estudiado
alpha1=alpha1_index(:,s);
SE_sidelength=singular_FE_sidelength_index(s);
file = file_index(s,:);
inp_malla=fileread(file);
tic
```

```
54
55 %% Obtenemos mallas
56 cd([ 'C:\Users\Usuario\Desktop\TFG (Avances y Bibliografia )\Macros\Codigo Matlab']);
57 [coordenadas,coordenadasN5,coordenadas5, triangulos , triangulos5 ]=step_04_mesher(inp_malla, nofelements, SE_sidelength,
       alpha1, typeelement, vct);
58
59 %% Imponer condiciones de contorno
60 [neumann,neumann5,dirichlet, dirichlet5 ,robin,robin5,nodos_calculo]=step_05_boundary_conditions(coordenadas, vct,
       triangulos , triangulos5 );
61
62 %% Contar elementos
63 [numNodos,numTriangulos,numTriangulos5,numLadosNeu,numLadosNeu5,numLadosDir,numLadosDir5,numLadosRob,
       numLadosRob5]=step_06_count_elements(coordenadas,triangulos,triangulos5,neumann,neumann5,dirichlet,dirichlet5,
       robin,robin5);
64
65 %% Matriz K_mn del elemento singular
66 RobMat=step_07_robin_matrix();
67
68 %% Inicializacion de variables
69 [K,b,KROBIN,u] = step_08_initialize_system_matrixes (numNodos);
70
71 %% Generacion de la matriz de rigidez del sistema
72 K = step_09_stiffness_matrix (coordenadas, triangulos , triangulos5 ,numTriangulos,numTriangulos5,G,K);
73
74 %% Generacion de las matrices que hay que ensamblar con la matriz de rigidez y que son debida a las condiciones
       robin
75 KROBIN = step_10_robin_stiffness_matrix(numLadosRob,numLadosRob5,robin,robin5,coordenadas,k_robin,RobMat,
       numTriangulos,triangulos,KROBIN,NUM);
76
77 %% ensamblaje de las condiciones robin en la matriz de rigidez :
78 K=K+KROBIN;
79
80 %% Condiciones neumann
81 b = step_11_neumann_conditions(b,numLadosNeu,numLadosNeu5,neumann,neumann5,coordenadas,numTriangulos,RobMat,
       numTriangulos5,triangulos,triangulos5);
82
83 %% condiciones dirichlet
84 [b,Dir,Dir5] = step_12_dirichlet_conditions (b,numLadosDir,numLadosDir5,dirichlet, dirichlet5 ,coordenadas,
       numTriangulos,RobMat,numTriangulos5,triangulos, triangulos5 );
85
86 %% vector independiente
87 b=b-K*u;
88
89 %% solucion
90 nodosDirTotales=[Dir;Dir5]; %forma una matriz con todos los nodos con condiciones Dirichlet
91 nodosLibres= setdiff (1:numNodos,nodosDirTotales); %excluye a los nodos con condiciones Dirichlet
92 u(nodosLibres)=K(nodosLibres,nodosLibres)\b(nodosLibres);
93 t=toc;
94 t_index=[t_index ; t ];
95
96 %% desplazamientos entorno a la grieta
97 desplazamientos_calculo =[ desplazamientos_calculo ; u(nodos_calculo) '];
98 coordenadas_calculo=[coordenadas_calculo ; coordenadas(nodos_calculo,1) '];
99 end
100
101 %% Ajuste
102 cd([ 'C:\Users\Usuario\Desktop\TFG (Avances y Bibliografia )\Macros\Codigo Matlab']);
103 [ ajuste , ajuste_2 , slope_index ,max_2nd_crown_change,casos_buenos,nota_a,nota_b, criterio_1 , criterio_2 , criterio_3a ,
       criterio_3b , criterio_4 ,criterio_1_num, criterio_1_den , criterio_5 , Coeff_vectora , Coeff_vectorb ,mean_slopes_index
       , criterio_2_mejorado ] = step_13_numerical_regressions (alpha1_index , alpha2_index , alpha3_index ,
       desplazamientos_calculo , coordenadas_calculo , tipo1 , tipo2 , id , separadores , coordenadas ,2.25596, t);
104
105 Coeff_vectora
106 Coeff_vectorb
107
108
109
110 if nofcrowns==2,
111     strvcat ([ 'p00=' num2str(Coeff_vectorb .p00,50) ],[ 'p10=' num2str(Coeff_vectorb .p10,50) ],[ 'p01=' num2str(
       Coeff_vectorb .p01,50) ],[ 'p20=' num2str(Coeff_vectorb .p20,50) ],[ 'p11=' num2str(Coeff_vectorb .p11,50) ],[ 'p02='
       num2str(Coeff_vectorb .p02,50) ])
```

```
else
    strvcat ([ 'p1=' num2str( ajuste_2 .p1,50) ],[ 'p2=' num2str( ajuste_2 .p2,50) ],[ 'p3=' num2str( ajuste_2 .p3,50) ])
    strvcat ([ 'p1=' num2str( ajuste .p1,50) ],[ 'p2=' num2str( ajuste .p2,50) ],[ 'p3=' num2str( ajuste .p3,50) ])
end


nota_index=[nota_index; nota_a, nota_b];

%% Guardado
cd([ 'C:\Users\Usuario\Desktop\TFG (Avances y Bibliografia )\Macros\', id (1: separadores (1)-1), '\' , id ]);
save(id);
cd([ 'C:\Users\Usuario\Desktop\TFG (Avances y Bibliografia )\Macros\Codigo Matlab']);
end
```

Script 9.3 Code to name and identify archives from Abaqus.

9.1.2 step 01

```
function [name_index, file_index , alpha1_index , alpha2_index , alpha3_index , desplazamientos_calculo , coordenadas_calculo ,
    singular_FE_sidelength_index , new_entrance_inp_index] = step_01_set_variables ()
name_index=[];
file_index =[];
alpha1_index =[];
alpha2_index =[];
alpha3_index =[];
desplazamientos_calculo =[];
coordenadas_calculo =[];
singular_FE_sidelength_index =[];
new_entrance_inp_index=[];
end
```

Script 9.4 Step 01 script.

step 02

```
function [nofelements, nofcrowns, tipo1 , tipo2 , separadores ] = step_02_obtain_parameters (id)
%% This code finds the crown type data by the naming code
separadores = strfind (id , '_');
nofelements=str2num(id (1: separadores (1)-2));
if str2num(id( separadores (end)+1:end))~=0
    nofcrowns=2;
else
    nofcrowns=1;
end
tipo1 =str2num(id( separadores (2)+1: separadores (3)-1));
tipo2 =str2num(id( separadores (4)+1:end));
```

Script 9.5 Step 02 script.

step 03

```
function [alpha1_index , alpha2_index , alpha3_index , singular_FE_sidelength_index , file_index , name_index,ncasos] =
    step_03_name_archives( nofelements , nofcrowns, tipo1 , tipo2 , hex_sidelength , id , name_index, file_index ,
    singular_FE_sidelength_index , alpha1_index , alpha2_index , alpha3_index , separadores )
cd([ 'C:\Users\Usuario\Desktop\TFG (Avances y Bibliografia )\Macros\', id (1: separadores (1)-1), '\' , id ]);

new_entrance=[];

%% 1 Corona:

%% tipo_1_tipo_0, tipo_2_tipo_0 , tipo_3_tipo_0 ,
if nofcrowns==1 && (tipo1==1 || tipo1==2 || tipo1==3)
    alpha1_in=0.05;
    alpha1_in_plus=0;
```

```
    alpha2_in_n=1/cos(pi/nofelements);
    alpha2_in_plus=0.05;
end
if  nofcrowns==1 && tipo1==4
    alpha1_in=0.05;
    alpha1_in_plus=0;
    alpha2_in_n=1/cos(pi/2/nofelements);
    alpha2_in_plus=0;
end
if  nofcrowns==1 && tipo1==5 && (nofelements==4 || nofelements==6 ||  nofelements==8)
    alpha1_in=0.05;
    alpha1_in_plus=0;
    alpha2_in_n=1/cos(pi/2/nofelements);
    alpha2_in_plus=0.0001;
end
if  nofcrowns==1 && tipo1==5 && (nofelements==12 || nofelements==24)
    alpha1_in=0.05;
    alpha1_in_plus=0;
    alpha2_in_n=1/cos(pi/2/nofelements);
    alpha2_in_plus=0.0001;
end

%% 2 Coronas:
if  nofcrowns==2 && (tipo1==1 ||  tipo1==2 ||  tipo1==3) && (tipo2==1 ||  tipo2==2 ||  tipo2==3)
    alpha1_in=0.05;
    alpha1_in_plus=0;
    alpha2_in_n=1/(cos(pi/nofelements));
    alpha2_in_plus=0.05;
    alpha3_in_n=1/(cos(pi/nofelements));
    alpha3_in_plus=0.05;
end

if  nofcrowns==2 && (tipo1==1 ||  tipo1==2 ||  tipo1==3) && tipo2==4
    alpha1_in=0.05;
    alpha1_in_plus=0;
    alpha2_in_n=1/(cos(pi/nofelements));
    alpha2_in_plus=0;
    alpha3_in_n=1/(cos(pi/2/nofelements));
    alpha3_in_plus=0;
end

if  nofcrowns==2 && (tipo1==1 ||  tipo1==2 ||  tipo1==3) && tipo2==5
    alpha1_in=0.05;
    alpha1_in_plus=0;
    alpha2_in_n=1/(cos(pi/nofelements));
    alpha2_in_plus=0.05;
    alpha3_in_n=1/(cos(pi/2/nofelements));
    alpha3_in_plus=0.05;
end

if  nofelements==4 && nofcrowns==2 && (tipo1==3) && tipo2==5
    alpha1_in=0.05;
    alpha1_in_plus=0;
    alpha2_in_n=1/(cos(pi/nofelements));
    alpha2_in_plus=0;
    alpha3_in_n=1/(cos(pi/2/nofelements));
    alpha3_in_plus=0.0001;
end

if  nofcrowns==2 && (tipo1==4) && (tipo2==1 ||  tipo2==2 ||  tipo2==3)
    alpha1_in=0.05;
    alpha1_in_plus=0;
    alpha2_in_n=1/(cos(pi/2/nofelements));
    alpha2_in_plus=0;
    alpha3_in_n=1/(cos(pi/nofelements));
    alpha3_in_plus=0.05;
end

if  nofcrowns==2 && (tipo1==4) && (tipo2==4)
    alpha1_in=0.05;
    alpha1_in_plus=0;
```

```
    alpha2_in_n=1/(cos(pi/2/nofelements));
    alpha2_in_plus=0;
    alpha3_in_n=1/(cos(pi/2/nofelements));
    alpha3_in_plus=0;
end

if  nofcrowns==2 && (tipo1==4) && (tipo2==5)
    alpha1_in=0.05;
    alpha1_in_plus=0;
    alpha2_in_n=1/(cos(pi/2/nofelements));
    alpha2_in_plus=0;
    alpha3_in_n=1/(cos(pi/2/nofelements));
    alpha3_in_plus=0;
end

if  nofcrowns==2 && (tipo1==5) && (tipo2==1 || tipo2==2 || tipo2==3) && (nofelements==4 || nofelements==6)
    alpha1_in=0.05;
    alpha1_in_plus=0;
    alpha2_in_n=1/(cos(pi/2/nofelements));
    alpha2_in_plus=0.0001;
    alpha3_in_n=1/(cos(pi/nofelements));
    alpha3_in_plus=0;
end

if  nofcrowns==2 && (tipo1==5) && (tipo2==1 || tipo2==2 || tipo2==3) && (nofelements==8)
    alpha1_in=0.05;
    alpha1_in_plus=0;
    alpha2_in_n=1/(cos(pi/2/nofelements));
    alpha2_in_plus=0.0001+0.05;
    alpha3_in_n=1/(cos(pi/nofelements));
    alpha3_in_plus=0.05;
end

if  nofcrowns==2 && (tipo1==5) && (tipo2==1 || tipo2==2 || tipo2==3) && (nofelements==12)
    alpha1_in=0.2;
    alpha1_in_plus=0;
    alpha2_in_n=1/(cos(pi/2/nofelements));
    alpha2_in_plus=0.0001+0.05;
    alpha3_in_n=1/(cos(pi/nofelements));
    alpha3_in_plus=0.05;
end

if  nofcrowns==2 && (tipo1==5) && (tipo2==1 || tipo2==2 || tipo2==3) && (nofelements==24)
    alpha1_in=0.2;
    alpha1_in_plus=0;
    alpha2_in_n=1/(cos(pi/2/nofelements));
    alpha2_in_plus=0.0001+0.05;
    alpha3_in_n=1/(cos(pi/nofelements));
    alpha3_in_plus=0;
end

if  nofcrowns==2 && (tipo1==5) && (tipo2==4) && nofelements~=12 && nofelements~=24,
    alpha1_in=0.05;
    alpha1_in_plus=0;
    alpha2_in_n=1/(cos(pi/2/nofelements));
    alpha2_in_plus=0.0001+0.05;
    alpha3_in_n=1/(cos(pi/2/nofelements));
    alpha3_in_plus=0.05;
end

if  nofcrowns==2 && (tipo1==5) && (tipo2==4) && (nofelements==12)
    alpha1_in=0.2;
    alpha1_in_plus=0;
    alpha2_in_n=1/(cos(pi/2/nofelements));
    alpha2_in_plus=0.0001;
    alpha3_in_n=1/(cos(pi/2/nofelements));
    alpha3_in_plus=0.05;
end

if  nofcrowns==2 && (tipo1==5) && (tipo2==4) && (nofelements==24)
    alpha1_in=0.2;
```

```
    alpha1_in_plus=0;
    alpha2_in_n=1/(cos(pi/2/nofelements));
    alpha2_in_plus=0.0001+0.05;
    alpha3_in_n=1/(cos(pi/2/nofelements));
    alpha3_in_plus=0.05;
end

if  nofcrowns==2 && (tipo1==5) && (tipo2==5) && nofelements~=12 && nofelements~=24,
    alpha1_in=0.05;
    alpha1_in_plus=0;
    alpha2_in_n=1/(cos(pi/2/nofelements));
    alpha2_in_plus=0.0001+0.05;
    alpha3_in_n=1/(cos(pi/2/nofelements));
    alpha3_in_plus=0.05;
end

if  nofcrowns==2 && (tipo1==5) && (tipo2==5) && (nofelements==12)
    alpha1_in=0.2;
    alpha1_in_plus=0;
    alpha2_in_n=1/(cos(pi/2/nofelements));
    alpha2_in_plus=0.0001+0.05;
    alpha3_in_n=1/(cos(pi/2/nofelements));
    alpha3_in_plus=0.05;
end

if  nofcrowns==2 && (tipo1==5) && (tipo2==5) && (nofelements==24)
    alpha1_in=0.2;
    alpha1_in_plus=0;
    alpha2_in_n=1/(cos(pi/2/nofelements));
    alpha2_in_plus=0.0001+0.05;
    alpha3_in_n=1/(cos(pi/2/nofelements));
    alpha3_in_plus=0.05;
end

%% One-crown
if  nofcrowns==1,
    for alpha1=alpha1_in+alpha1_in_plus:0.05:0.85
        for alpha2=alpha1*alpha2_in_n+alpha2_in_plus:0.05:0.85
            singular_FE_sidelength=alpha1*hex_sidelength;
            alpha1_s=num2str((alpha1),4);
            alpha1_s=strrep(alpha1_s,'.','');
            alpha2_s=num2str((alpha2),'%2.15f');
            alpha2_s=strrep(alpha2_s,'.','');
            alpha2_s=alpha2_s(1:4);
            Files=dir(['*','_',alpha1_s,'_',alpha2_s(1:end),'*.inp']);
            if isempty(Files)
                alpha2_s=strrep(num2str(alpha2+0.0001,'%2.15f'),'.','');
                alpha2_s=alpha2_s(1:4);
                Files=[];
                Files=dir(['*','_',alpha1_s,'_',alpha2_s(1:end),'*.inp']);
                if isempty(Files)
                    alpha2_s=strrep(num2str(alpha2-0.0001,'%2.15f'),'.','');
                    alpha2_s=alpha2_s(1:4);
                end
            end
            new_entrance=[id alpha1_s '_' alpha2_s];
            new_entrance_inp=[id '_' alpha1_s '_' alpha2_s '.inp'];
            name_index=strvcat(name_index,new_entrance);
            file_index=strvcat(file_index,new_entrance_inp);
            singular_FE_sidelength_index=[singular_FE_sidelength_index; singular_FE_sidelength];
            alpha1_index=[alpha1_index,alpha1];
            alpha2_index=[alpha2_index,alpha2];
        end
    end
end

%% Two-crowns
if  nofcrowns==2
    for alpha1=alpha1_in+alpha1_in_plus:0.05:0.85
        for alpha2=alpha1*alpha2_in_n+alpha2_in_plus:0.05:0.85
            for alpha3=alpha2*alpha3_in_n+alpha3_in_plus:0.05:0.85
```

```
% Singular element side-length
singular_FE_sidelength =alpha1*hex_sidelength;

% alpha1
alpha1_s=num2str((alpha1),4);
alpha1_s= strrep (alpha1_s,'.','');

% alpha2
alpha2_s=num2str((alpha2),'%2.15f');
alpha2_s= strrep (alpha2_s,'.','');
alpha2_s=alpha2_s(1:4);

% alpha3
alpha3_s=num2str((alpha3),'%2.15f');
alpha3_s= strrep (alpha3_s,'.','');
if length(alpha3_s)>3, alpha3_s=alpha3_s(1:4); else, alpha3_s=alpha3_s(1:end); end

% = alpha2, = alpha3
Files =dir ([ '*','_',alpha1_s,'_',alpha2_s(1:end),'_',alpha3_s(1:end),'*.inp' ]);
if isempty( Files ),
    % = alpha2, + alpha3.
    alpha2_aux=str2num([alpha2_s(1),'.',alpha2_s(2:end)]);
    alpha2_saux= strrep (num2str(alpha2_aux+0.001,'%2.15f'),'.','');
    alpha3_aux=str2num([alpha3_s(1),'.',alpha3_s(2:end)]);
    alpha3_saux= strrep (num2str(alpha3_aux+0.001,'%2.15f'),'.','');
    alpha3_saux=alpha3_saux(1:4);
    Files =dir ([ '*','_',alpha1_s,'_',alpha2_saux(1:end),'_',alpha3_saux(1:end),'*.inp' ]);
    if ~isempty( Files ),
        alpha2_s=alpha2_saux;
        alpha3_s=alpha3_saux;
    else
    % = alpha2, -alpha3.
    alpha2_aux=str2num([alpha2_s(1),'.',alpha2_s(2:end)]);
    alpha3_aux=str2num([alpha3_s(1),'.',alpha3_s(2:end)]);
    alpha3_saux= strrep (num2str(alpha3_aux-0.001,'%2.15f'),'.','');
    alpha3_saux=alpha3_saux(1:4);
    Files =dir ([ '*','_',alpha1_s,'_',alpha2_saux(1:end),'_',alpha3_saux(1:end),'*.inp' ]);
    if ~isempty( Files ),
        alpha2_s=alpha2_saux;
        alpha3_s=alpha3_saux;
    else
    % + alpha2, = alpha3
    alpha2_aux=str2num([alpha2_s(1),'.',alpha2_s(2:end)]);
    alpha3_aux=str2num([alpha3_s(1),'.',alpha3_s(2:end)]);
    alpha2_saux= strrep (num2str(alpha2_aux+0.001,'%2.15f'),'.','');
    alpha2_saux=alpha2_saux(1:4);
    Files =dir ([ '*','_',alpha1_s,'_',alpha2_saux(1:end),'_',alpha3_saux(1:end),'*.inp' ]);
    if ~isempty( Files ),
        alpha2_s=alpha2_saux;
        alpha3_s=alpha3_saux;
    else
    % + alpha2, + alpha3
    alpha2_aux=str2num([alpha2_s(1),'.',alpha2_s(2:end)]);
    alpha3_aux=str2num([alpha3_s(1),'.',alpha3_s(2:end)]);
    alpha2_saux= strrep (num2str(alpha2_aux+0.001,'%2.15f'),'.','');
    alpha2_saux=alpha2_saux(1:4);
    alpha3_saux= strrep (num2str(alpha3_aux+0.001,'%2.15f'),'.','');
    alpha3_saux=alpha3_saux(1:4);
    Files =dir ([ '*','_',alpha1_s,'_',alpha2_saux(1:end),'_',alpha3_saux(1:end),'*.inp' ]);
    if ~isempty( Files ),
        alpha2_s=alpha2_saux;
        alpha3_s=alpha3_saux;
    else
    % + alpha2, - alpha3
    alpha2_aux=str2num([alpha2_s(1),'.',alpha2_s(2:end)]);
    alpha3_aux=str2num([alpha3_s(1),'.',alpha3_s(2:end)]);
    alpha2_saux= strrep (num2str(alpha2_aux+0.001,'%2.15f'),'.','');
    alpha2_saux=alpha2_saux(1:4);
    alpha3_saux= strrep (num2str(alpha3_aux-0.001,'%2.15f'),'.','');
    alpha3_saux=alpha3_saux(1:4);
    Files =dir ([ '*','_',alpha1_s,'_',alpha2_saux(1:end),'_',alpha3_saux(1:end),'*.inp' ]);
```

```
                    if ~isempty(Files),
                        alpha2_s=alpha2_saux;
                        alpha3_s=alpha3_saux;
                    else
                    % - alpha2, = alpha3
                    alpha2_aux=str2num([alpha2_s(1),'.',alpha2_s(2:end)]);
                    alpha3_aux=str2num([alpha3_s(1),'.',alpha3_s(2:end)]);
                    alpha2_saux=strrep(num2str(alpha2_aux-0.001,'%2.15f'),'.','');
                    alpha2_saux=alpha2_saux(1:4);
                    Files=dir(['*','_',alpha1_s,'_',alpha2_saux(1:end),'_',alpha3_saux(1:end),'*.inp']);
                    if ~isempty(Files),
                        alpha2_s=alpha2_saux;
                        alpha3_s=alpha3_saux;
                    else
                    % - alpha2, + alpha3
                    alpha2_aux=str2num([alpha2_s(1),'.',alpha2_s(2:end)]);
                    alpha3_aux=str2num([alpha3_s(1),'.',alpha3_s(2:end)]);
                    alpha2_saux=strrep(num2str(alpha2_aux-0.001,'%2.15f'),'.','');
                    alpha2_saux=alpha2_saux(1:4);
                    alpha3_saux=strrep(num2str(alpha3_aux+0.001,'%2.15f'),'.','');
                    alpha3_saux=alpha3_saux(1:4);
                    Files=dir(['*','_',alpha1_s,'_',alpha2_saux(1:end),'_',alpha3_saux(1:end),'*.inp']);
                    if ~isempty(Files),
                        alpha2_s=alpha2_saux;
                        alpha3_s=alpha3_saux;
                    else
                    % - alpha2, - alpha3
                    alpha2_aux=str2num([alpha2_s(1),'.',alpha2_s(2:end)]);
                    alpha3_aux=str2num([alpha3_s(1),'.',alpha3_s(2:end)]);
                    alpha2_saux=strrep(num2str(alpha2_aux-0.001,'%2.15f'),'.','');
                    alpha2_saux=alpha2_saux(1:4);
                    alpha3_saux=strrep(num2str(alpha3_aux-0.001,'%2.15f'),'.','');
                    alpha3_saux=alpha3_saux(1:4);
                    Files=dir(['*','_',alpha1_s,'_',alpha2_saux(1:end),'_',alpha3_saux(1:end),'*.inp']);
                    if ~isempty(Files),
                        alpha2_s=alpha2_saux;
                        alpha3_s=alpha3_saux;
                    end
                    end
                    end
                    end
                    end
                    end
                    end
                    end
                end
                if isempty(Files)
                    new_entrance_inp=[];
                    singular_FE_sidelength_index=[singular_FE_sidelength_index ; []];
                    name_index=strvcat(name_index, new_entrance);
                    alpha1_index=[alpha1_index ,[]];
                    alpha2_index=[alpha2_index ,[]];
                    alpha3_index=[alpha3_index ,[]];
                else
                    new_entrance_inp=[id '_' alpha1_s '_' alpha2_s '_' alpha3_s '.inp'];
                    name_index=strvcat(name_index, new_entrance);
                    file_index = strvcat( file_index ,new_entrance_inp);
                    singular_FE_sidelength_index=[singular_FE_sidelength_index ; singular_FE_sidelength ];
                    alpha1_index=[alpha1_index,alpha1 ];
                    alpha2_index=[alpha2_index,alpha2 ];
                    alpha3_index=[alpha3_index,alpha3 ];
                end
            end
        end
    end
end

ncasos=length( file_index );
```

Script 9.6 Step 03 Script.

9.1.3 step 04

```
function [coordenadas,coordenadasN5,coordenadas5, triangulos , triangulos5 ] = step_04_mesher(inp_malla, nofelements,
    sidelength , alpha, typeelement, vct);
xcentro=vct(1); ycentro=vct(2); zcentro=vct(3);

%% Matriz de coordenadas
begin_reading_coordenadas= strfind (inp_malla, '*Node'); % Leo desde "*Node".
end_reading_coordenadas= strfind (inp_malla, '*Element'); % Hasta "*Element".
coordenadas_aux=[inp_malla(begin_reading_coordenadas+8:end_reading_coordenadas-2)]; % Saco las coordenadas del
    documento. El +5 es para leer el inicio de los números, y no el encabezado. El -2 es para no coger los
    caracteres del encabezado del siguiente punto.
coordenadas_aux= strsplit (coordenadas_aux,'\n'); % Corto el texto por líneas según el caracter "salto de línea".
coordenadas_aux=coordenadas_aux'; % Ordeno el texto en vertical .
for i=1:length(coordenadas_aux)
    coordenadas(i,:) =str2num(cell2mat(coordenadas_aux(i))); % Convierto cada fila en matriz de texto, y cada matriz
    de texto (cell) en matriz numérica.
end
coordenadas (:,1) =[]; % Elimino la columna del número de identificaci ón del nodo.

%% Matriz de conectividad de elementos

begin_reading_triangulos = strfind (inp_malla, '*Element,'); % Se encuentra el comienzo de la matriz de tri ángulos
end_reading_triangulos = strfind (inp_malla, '*End Part'); % Se encuentra el final de la matriz de tri ángulos
triangulos_aux =[inp_malla( begin_reading_triangulos +21: end_reading_triangulos -2)]; % Se introducen los valores en
    una variable
triangulos_aux = strsplit ( triangulos_aux , '\n'); % Se divide las líneas de la matriz por el caracter de salto de línea
triangulos_aux =triangulos_aux '; % Se pone en el formato correcto
for i=1:length( triangulos_aux )
    triangulos (i ,:) =str2num(cell2mat( triangulos_aux (i))); % Se convierte la línea de la variable tri ángulos en nú
    meros
end
triangulos (:,1) =[]; % Se elimina la primera columna, que es innecesaria

%% Matriz de coordenadas de los elementos singulares

N=2*nofelements; % El número de lados del polígono regular cuya mitad superior ser ían los elementos singulares .
n=(0:nofelements);
r=sidelength *1/2;
if typeelement==1 % Si existen elementos singulares
    x(n+1)=r*cos(2*pi*n/N);
    y(n+1)=r*sin(2*pi*n/N); % Calculo las coordenadas x, y del polígono regular .
    z(n+1)=zcentro; % Introducir la z pertinente .
    coordenadas_singulares =[xcentro, ycentro, zcentro; x', y', z'];
elseif typeelement==0 % Si no existen los elementos singulares
    coordenadas_singulares =[xcentro, ycentro, zcentro];
end
coordenadas_singulares (:,3) =[]; % Se elimina la tercera columna, innecesaria

%% Matriz de conectividad de los elementos singulares

r=sidelength ;
x(n+1)=r*cos(2*pi*n/N);
y(n+1)=r*sin(2*pi*n/N);
z(n+1)=0; % Introducir la z pertinente .
for n=1:nofelements+1
    closestpoint (n)=dsearchn(coordenadas,[x(n) y(n)]); % Encuentro el punto del listado más próximo al obtenido de
    nuestra fórmula
end
if typeelement==1 % Se crea la matriz de conectividad de los elementos singulares . Los puntos est án en el sentido
    de las agujas del reloj: el primer nodo es el central, el segundo nodo es el siguiente al central, el tercero
    es el punto que ya exist ía previamente en la malla (el final del elemento singular), el cuarto el punto
    siguiente al anterior (que también preexiste en la malla), y el último el siguiente punto intermedio.
    for i=1:nofelements
        triangulos5 (i ,:) =[size(coordenadas,1)+1 size(coordenadas,1)+1+i closestpoint (i) closestpoint (i+1) size(
    coordenadas,1)+2+i];
    end
elseif typeelement==0
    for i=1:nofelements % Aquí no hay puntos intermedios, solo tres
        triangulos5 (i ,:) =[size(coordenadas,1)+1 closestpoint (i) closestpoint (i+1)];
```

```
    end
    triangulos =[ triangulos ; triangulos5 ];
    triangulos5 =[];
end
%% Terminamos de ensamblar la matriz de coordenadas
coordenadas5=coordenadas_singulares;
coordenadasN5=coordenadas;
coordenadas=[coordenadas; coordenadas_singulares ];
```

Script 9.7 Step 03 Script.

step 13

```
function [ ajuste , ajuste_2 , slope_index , max_2nd_crown_change,casos_buenos,nota_a,nota_b, criterio_1 , criterio_2 ,
    criterio_3a , criterio_3b , criterio_4 , criterio_1_num , criterio_1_den , criterio_5 , Coeff_vectora , Coeff_vectorb ,
    mean_slopes_index, criterio_2_mejorado ] = step_13_numerical_regressions (alpha1_index , alpha2_index , alpha3_index
    , desplazamientos_calculo , coordenadas_calculo , tipo1 , tipo2 , id , separadores , coordenadas, rcts , t)
cd([ 'C:\Users\Usuario\Desktop\TFG (Avances y Bibliografia )\Macros\', id (1: separadores (1)-1), '\', id ]);
% clc

%% X: alpha_2, Y: alpha_3, Z: u_ct; X coordinate: alpha_2; Y coordinate: alpha_3; Z coordinate: u_ct

%% Variables initialization
leyenda =[]; ajuste =[];
x_pr=[]; y_pr=[]; z_pr=[];
casos_buenos=[]; slope_index =[]; level_curve =[]; point_vector =[]; max_2nd_crown_change=[]; mean_slopes_index=[];
    slope_index_aux=[]; criterio_2_mejorado =[];
change=[1]; change2=[1];

%% This section fills sparce matrices (FEM results).
desplazamientos_calculo = full ( desplazamientos_calculo );
coordenadas_calculo= full ( coordenadas_calculo );

%% This section finds where the indices change, that is, all the cases with the same alpha_1 value.


for s=1:length (alpha1_index)-1
    if alpha1_index(s)-alpha1_index(s+1)~=0 % Each value that does not coincide with the following
        change=[change, s+1]; %      its position is added as the first unequal values .
    end
end

%% For the one-crown cases:
if tipo2==0

    %% Finding an optimal point. We must find the right proportion between alpha_1 and alpha_2 that results in an
      u_ct value of 2.25596. To do that, we plot u_ct vs alpha_2 for each alpha_1 (there will be several curves).
      If we also plot u_ct=2.25596 we can find a cut-off point between curves (since they are defined by points, we
       will be using a spline to find the cut-off point). For each alpha_1 value we find the alpha_2 and alpha_3
      values that gives the right u_ct. A simple numerical correlation is presupposed.
    % For one-crown cases, we can scatter alpha_1 vs optimal alpha_2 values. Then, a ployfit regression is made.

    for s2=1:length (change)-1 % For each curve (there are as many curves as changes in vector alpha_1) taken as an
      entry of the vector change

        %%    The crack tip displacements are plotted
        alpha1_legend=num2str(alpha1_index(change(s2)));
        % First figure: u_ct vs alpha_2 for each alpha_1
        figure (1)
        xlabel ('\alpha_2'); ylabel ('u_1');
        plot (alpha2_index (change(s2):change(s2+1)-1), desplazamientos_calculo (change(s2):change(s2+1)-1,1), '
      linewidth ',2)
        if s2~=length(change)-1
            leyenda= strvcat (leyenda ,[ '\alpha_1=', alpha1_legend ]);
        end
        hold on

        %% This section calculates the intersection between u_ct and alpha_1. A spline is generated with the
      previously calculated points (the FEM results).
```

```
46          y_dado=desplazamientos_calculo(change(s2):change(s2+1)-1,1); % displacement values
47          x_dado=alpha2_index(change(s2):change(s2+1)-1); % alpha_2 values (corresponding to the values between the í
        ndices of alpha_1 changes).
48          desiredy = 2.25596; % Obtained by a very precise calculation with fine meses by [FALTA DECIR QUIÉN]
49          fun = @(x) desiredy - spline(x_dado,y_dado,x); % We must find a zero of this function: the spline value
        must be equal to the desired value.
50          x=sym('x','positive');
51          desiredx = fzero(fun, 0.3);
52          Interseccion_alpha2(s2) = desiredx;
53          longitud_el_sing(s2) = (alpha1_index(change(s2)));
54          longitud_el_cor(s2) = (desiredx-alpha1_index(change(s2)));
55
56
57          %% Slope calculation. The slopes at the cut-off point can be used as mesh quality index as explained in the
        following section.
58          for s3=1:length(x_dado)-1; % Among the values of alpha_2 from the curve
59              if (desiredx>x_dado(s3) && x_dado(s3+1)>desiredx), % we find the values between which the cutt-off is.
60                  point_vector=[point_vector; desiredx, desiredy];
61                  index=s3;
62                  casos_buenos=[casos_buenos; alpha1_index(change(s2))]; % We only consider the case as valid if the
        cut-off is found by interpolation instead of extrapolation. Out of the computed values, in general there is
        no geometrically feasible crown: alpha_2 would be either smaller than alpha_1 or bigger than the half-
        hexagon.
63                  slope=(y_dado(index+1)-y_dado(index))/(x_dado(index+1)-x_dado(index)); % From the previous data,
        the segments tangent slope is calculated and introduce in an index.
64                  slope_index=[slope_index;slope];
65              end
66          end
67      end
68      scatter(point_vector(:,1),point_vector(:,2));
69
70      %% Adjustment (linear regression).
71  %       [ajuste,gofa]=fit(longitud_el_sing(floor(casos_buenos*20))',longitud_el_cor(floor(casos_buenos*20))','poly1')
        ; % A linear adjustment
72      [ajuste,gofa]=fit((longitud_el_sing(floor(casos_buenos*20))+longitud_el_cor(floor(casos_buenos*20)))',
        longitud_el_sing(floor(casos_buenos*20))','poly2');
73      [ajuste_2,gofb]=fit(longitud_el_sing(floor(casos_buenos*20))',longitud_el_cor(floor(casos_buenos*20))','poly2')
        ; % And a parabolic adjustment
74
75      Coeff_vectora=ajuste;
76      Coeff_vectorb=ajuste_2;
77
78      % Posterior adición
79      Coeff_vectora=fit(longitud_el_sing(floor(casos_buenos*20))',longitud_el_cor(floor(casos_buenos*20))','poly1');
80  %       Coeff_vectora=[Coeff_vectora.p1-1 Coeff_vectora.p2];
81
82
83      s2=s2+1;
84      %% Plot curves u_ct vs alpha_1
85      plot(alpha2_index(change(s2):end),desplazamientos_calculo(change(s2):end,1),'linewidth',2)
86      leyenda=strvcat(leyenda,+['\alpha_1=',num2str(alpha1_index(end))]);
87      legend(leyenda)
88      xlabel('\alpha_2')
89      ylabel('u_1')
90      xlim([0.05 1.1])
91      hold on
92      plot([0:1],[2.25596,2.25596])
93      set(gca,'XAxisLocation','origin','YAxisLocation','origin')
94      print('-dpng','-r300',['graph_curvas_uvsalpha2',id,'.png'])
95
96      %% Adjustment curve plot
97      % Second figure: scattered points vs 1st degree adjustment
98      figure(2); xlabel('\alpha_1'), ylabel('\alpha_2-\alpha_1')
99      ajuste_lin=polyfit(longitud_el_sing(floor(casos_buenos*20)),longitud_el_cor(floor(casos_buenos*20)),1);
100     scatter(longitud_el_sing(floor(casos_buenos*20)),longitud_el_cor(floor(casos_buenos*20))); hold on;
101     x_ajuste_lin=[longitud_el_sing(floor(casos_buenos(1)*20)):0.01:longitud_el_sing(floor(casos_buenos(end)*20))];
102     y_ajuste_lin=ajuste_lin(1).*x_ajuste_lin+ajuste_lin(2);
103     plot(x_ajuste_lin,y_ajuste_lin,'Linewidth',2); hold on;
104     set(gca,'XAxisLocation','origin','YAxisLocation','origin')
105     print('-dpng','-r300',['graph_curvas_ajuste_',id,'.png'])
106     % Third figure: scattered points vs 2nd degree adjustment
```

```
107     figure (3)
108     ajuste_par = polyfit ( longitud_el_sing ( floor (casos_buenos*20)), longitud_el_cor ( floor (casos_buenos*20)),2) ; %
          Normal
109     ajuste_par = polyfit ( longitud_el_cor ( floor (casos_buenos*20))+ longitud_el_sing ( floor (casos_buenos*20)),
          longitud_el_sing ( floor (casos_buenos*20)),2) ; % Alternativo
110     scatter ( longitud_el_sing ( floor (casos_buenos*20)), longitud_el_cor ( floor (casos_buenos*20))); hold on;
111     x_ajuste_par =[ longitud_el_sing (round(casos_buenos(1)*20)) :0.01: longitud_el_sing (round(casos_buenos(end)*20)) ];
112     y_ajuste_par = ajuste_par (1) .* x_ajuste_par .^2+ ajuste_par (2) .* x_ajuste_par + ajuste_par (3) ;
113     plot ( x_ajuste_par , y_ajuste_par ); hold on;
114     set (gca, 'XAxisLocation', ' origin ', 'YAxisLocation', ' origin ')
115     print ('-dpng', '-r300',[' graph_curvas_ajuste_todos_ ', id , '.png'])
116
117     %% Score
118     if isempty(casos_buenos)==0
119         criterio_1 = length (casos_buenos)/ length (alpha1_index(change));
120         criterio_1_num=casos_buenos; criterio_1_den =alpha1_index(change);
121         theta = atan(mean(slope_index));
122         criterio_2 = (pi/2+ theta )/( pi /2) ;
123         x = longitud_el_sing ( floor (casos_buenos*20)); % Create x
124         y = longitud_el_cor ( floor (casos_buenos*20)); % Create y
125         Rsqa = gofa . rsquare ;
126         Rsqb = gofb . rsquare ;
127         criterio_3a =Rsqa;
128         criterio_3b =Rsqb;
129         criterio_4 =1-size(coordenadas,1) /1131;
130         criterio_5 =mean(t);
131         nota_a=( criterio_1 + criterio_2 + criterio_3a + criterio_4 ) /4;
132         nota_b=( criterio_1 + criterio_2 + criterio_3b + criterio_4 ) /4;
133     else
134         criterio_1 =0; criterio_2 =0; criterio_3a =0; criterio_3b =0; criterio_4 =0;
135         nota_a=0; nota_b=0;
136     end
137 end
138
139 %% For two-crown cases
140 if tipo2~=0
141
142     %% This section finds the optimal curve. Now there are three axis: alpha_2, alpha_3 and u_ct. The surface that
          relates those parameters for every alpha_1. When that surface is cut by a plane with constant value u_ct
          =2.25596 we obtain a level curve from which we can extract to values. The first one, the variation in the
          solution produced by the second crown. The second one, a relationship among optimal values for alpha_1,
          alpha_2 and alpha_3 (as an adjustment plane).
143     for s2=1:length(change)-1
144         desiredz = 2.25596;
145         alpha1_pr=alpha1_index(change(s2));
146
147         %% This section interpolates the surface points .
148         X=alpha2_index(change(s2):change(s2+1)-1) ';
149         Y=alpha3_index(change(s2):change(s2+1)-1) ';
150         Z=desplazamientos_calculo(change(s2):change(s2+1)-1,1) ';
151         x = X; y = Y; z = Z;
152
153         %% This section identifies if the case is valid or not. Since the surface is continuous, if its maximum
            value is superior to the desired one and its minimal value is below the desired value, the surface will acquire
            the desired value at some point.
154         if max(Z(:))> desiredz && min(Z(:))<desiredz
155             casos_buenos=[casos_buenos; alpha1_index(change(s2)) ];
156         end
157
158         %% Surface points are also interpolated .
159         bound=boundary(x,y); % This line finds the contour of the Surface projected on the z=cte plane.
160         for s3=1:length(Y) % From the previous tests we have observed that the level curve y vs x is biyective , but
            x vs y is not. For this process, we must do as follows: we generate a spline with sparce values from the
            previously calculated meshes (u_ct, alpha_2, alpha_3). For each value of alpha_3 (y) from the index, we use
            the interpolating function maintaining the y value constant while keeping x_interp as a symbolic variable (
            representing alpha_2). The x cut-off value of our surface with the u_ct-constant plane for the fixed y is
            found by the function fzero.
161             y_interp=Y(s3);
162             F = scatteredInterpolant (X,Y,Z');
163             fun = @(x_interp) F(x_interp , y_interp ) - desiredz ;
164             x_interp=sym('x_interp ',' positive ');
```

```
165             desiredx = fzero(fun, 0.5);
166             if inpolygon(desiredx, y_interp, x(bound), y(bound))==1, % The following code will only be executed if the cut-off point is inside the proyected surface, otherwise the case will not be valid since the relationship between alpha_1, alpha_2 and alpha_3 is not allowed (due to geometric limitations).
167                 level_curve=[level_curve; desiredx, y_interp, desiredz]; % If it is valid, this point is added to the level curve. This vector will define the x, y, z coordinates of the intersection curve.
168                 %%
169                 x_dado=alpha2_index(change(s2):change(s2+1)-1);
170                 y_dado=desplazamientos_calculo(change(s2):change(s2+1)-1,1);
171                 for s3=1:length(x_dado)-1; % Same process as before
172                     if (desiredx>x_dado(s3) && x_dado(s3+1)>desiredx),
173                         index=s3;
174                         slope=(y_dado(index+1)-y_dado(index))/(x_dado(index+1)-x_dado(index));
175                         slope_index=[slope_index; slope];
176                         slope_index_aux=[slope_index_aux; slope];
177                     end
178                 end
179                 % First figure: level curve points and 3D curve of results
180                 % (that should be contained within the surface).
181                 figure(1)
182                 hold on
183                 scatter3(level_curve(:,1), level_curve(:,2), level_curve(:,3), 'Linewidth',3) % Curve points are plotted.
184
185
186             end
187         end
188         hold on
189         plot3(X,Y,Z) % A 3D-curve is plotted by the points resulting from simulations.
190         hold on
191
192         %% Level curve adjustment (with the axis changed to fit a parabolic distribution).
193         if isempty(level_curve)==0 % If the level curve exists
194             ajuste_entry = polyfit(level_curve(:,2), level_curve(:,1),2); % The level curve data are adjusted.
195             ajuste=[ajuste; ajuste_entry];
196             x_pr=[x_pr; level_curve(:,1)]; % Values are introduced
197             y_pr=[y_pr; level_curve(:,2)];
198             % More precisely: for each alpha_1, the relationship between alpha_2 y alpha_3 that produces the right u_ct is not very dependant on alpha_3. Said parabolas (that produced the adjustment) are very close to straight lines. For a fixed alpha_2 value, the difference between maximum and minimum u_ct values (that depend only on alpha_3) is stored.
199             z_pr=[z_pr; alpha1_pr*ones(size(level_curve(:,1)))]; % The maximal variation produced by the second crown is stored.
200             level_curve=[];
201         end
202
203         %% Maximal curve variation
204         alpha2_alpha1_cte=[]; uct_alpha_1_cte=[]; change2=[];
205         alpha2_alpha1_cte=alpha2_index(change(s2):change(s2+1));
206         uct_alpha_1_cte=desplazamientos_calculo(change(s2):change(s2+1));
207         for s3=1:length(alpha2_alpha1_cte)-1
208             if alpha2_alpha1_cte(s3)-alpha2_alpha1_cte(s3+1)~=0 % Same selection process
209                 change2=[change2, s3+1];
210             end
211         end
212         for s4=1:length(change2)-1, % For each and every repeated alpha_2 value, the subtraction between maximum and minimum u_ct values is stored.
213             alpha2_cte=alpha2_alpha1_cte(change2(s4):change2(s4+1)-1);
214             uct_alpha_2_cte=uct_alpha_1_cte(change2(s4):change2(s4+1)-1);
215             max_2nd_crown_change=[max_2nd_crown_change;abs(max(uct_alpha_2_cte)-min(uct_alpha_2_cte))];
216         end
217
218         %% Curve plot
219         xv = linspace(min(x), max(x), 20); % Linear alpha_2 interpolation values
220         yv = linspace(min(y), max(y), 20); % Linear alpha_3 interpolation values
221         [X,Y] = meshgrid(xv, yv); % X-Y reticule generation
222         Z = griddata(x,y,z,X,Y); % Sparce data interpolation to fit the Surface.
223         surf(X,Y,Z)
224
225         if ~isempty(slope_index_aux)==1
226             mean_slopes_index=[mean_slopes_index;mean(slope_index_aux)];
```

```
227                 slope_index_aux=[];
228             end
229
230         end
231
232         %% Ajuste de los planos
233         Xcolv = x_pr(:); % alpha_2 from the level curve
234         Ycolv = y_pr(:); % alpha_3 from the level curve
235         Zcolv = z_pr(:); % alpha_1 from the alpha_1 index
236         Const = ones(size(Xcolv));
237         Coefficients = [Xcolv Ycolv Const]\Zcolv; % This section computes the plane coefficients by the Least Squares
              Method.
238         XCoeff = Coefficients(1);
239         YCoeff = Coefficients(2);
240         CCoeff = Coefficients(3);
241
242         %% alpha_1-alpha_2-alpha_3 adjustment.
243         % Third figure: fitting planes (alpha_1 vs alpha_3 vs alpha_3)
244         figure(2)
245         L=plot3(x_pr,y_pr,z_pr,'ro'); % Plot the original data points
246         set(L,'Markersize',2*get(L,'Markersize')) % Making the circle markers larger
247         set(L,'Markerfacecolor','r') % Filling in the markers
248         hold on
249         [xx, yy]=meshgrid(0:0.1:1,0:0.1:1); % Generating a regular grid for plotting
250         zz = XCoeff * xx + YCoeff * yy + CCoeff;
251         surf(xx,yy,zz) % Plotting the surface
252         title(sprintf('z=(%f)*x+(%f)*y+(%f)',XCoeff, YCoeff, CCoeff))
253         xlabel('\alpha_2'); ylabel('\alpha_3'); zlabel('\alpha_1');
254         print('-dpng', '-r300', ['graph_plano_ajuste_',id,'_.png'])
255         ajuste_2=[-XCoeff -YCoeff 1 CCoeff]; %% Z coordinate pointing upwards
256
257         %% MATLAB Surface fit
258         Coeff_vectora=[0 0]; Rsquare_plane_a=0;
259         Coeff_vectorb=[0 0 0]; Rsquare_plane_b=0;
260         if size(x_pr,1)>=3
261             [Coeff_vectora,gofa]= fit([x_pr, y_pr],z_pr,'poly11');
262
263             %posterior adición
264             [Coeff_vectora,gofa]= fit([z_pr, y_pr],x_pr,'poly11');
265
266             Rsquare_plane_a=gofa.rsquare;
267         end
268         if size(x_pr,1)>6
269             [Coeff_vectorb,gofb]= fit([x_pr, y_pr],z_pr,'poly22');
270 %           [Coeff_vectorb,gofb]= fit([z_pr, y_pr],x_pr,'poly22'); % Normal method
271             [Coeff_vectorb,gofb]= fit([x_pr, y_pr],z_pr,'poly22'); % Alternate method
272             Rsquare_plane_b=gofb.rsquare;
273         end
274
275
276         %% Right-solution-plane graphic
277         % First and second figures: surfaces (u_ct vs alpha_2 vs alpha_3 for each alpha_1)
278         figure(1)
279         grid on
280         xlabel('\alpha_2')
281         zlabel('u_1')
282         ylabel('\alpha_3')
283         hold on
284         [X,Y] = meshgrid(0:0.7:0.7,0.1:0.8:0.9);
285         Z = 2.25596*ones(size(X)); view(45,20)
286         surf(X,Y,Z)
287         axis([0 0.8 0 1 1.75 2.5])
288         print('-dpng', '-r300', ['graph_curva_ajuste_',id,'.png']);
289         view([0 0 1])
290         print('-dpng', '-r300', ['graph_curva_ajuste_arriba_',id,'.png'])
291
292         %% Marking
293         if isempty(casos_buenos)==0
294             criterio_1 = length(casos_buenos)/length(alpha1_index(change));
295             criterio_1_num=casos_buenos; criterio_1_den=alpha1_index(change);
296             theta = atan(mean(slope_index));
```

```
            criterio_2  = (pi/2+theta)/(pi/2);
            criterio_2_mejorado =(pi/2+atan(mean(mean_slopes_index)))/(pi/2);
            criterio_3a  = Rsquare_plane_a;
            criterio_3b  = Rsquare_plane_b;
            criterio_4 =1-size(coordenadas,1)/1131;
            nota_a=( criterio_1 + criterio_2 + criterio_3a + criterio_4 )/4;
            nota_b=( criterio_1 + criterio_2 + criterio_3b + criterio_4 )/4;
            criterio_5 =mean(t);
        end
        if isempty(casos_buenos)==1
            criterio_1 =0;  criterio_2 =0;  criterio_3a =0;  criterio_3b =0;  criterio_4 =0;
            nota_a=0; nota_b=0;
        end
end
[nota_a nota_b];
% close all
cd([ 'C:\Users\Usuario\Desktop\TFG (Avances y Bibliografia )\Macros\Codigo Matlab']);
```

Script 9.8 Step 13 script.

9.1.4 MATLAB Macros for the second study

Main steps are the same, only the third and last one change.

step 03

```
function [alpha1_index,alpha2_index,alpha3_index, singular_FE_sidelength_index , file_index ,name_index,ncasos,m] =
        step_03_name_archives(nofelements,nofcrowns,tipo1 , tipo2 , hex_sidelength , id ,name_index, file_index ,
         singular_FE_sidelength_index ,alpha1_index,alpha2_index,alpha3_index, separadores ,method)
cd([ 'C:\Users\Usuario\Desktop\TFG (Avances y Bibliografia )\Macros 2\' ,id(1: separadores(1)-1), '\' ,id]);

new_entrance=[];

%% 1 Corona:

%% tipo_1_tipo_0, tipo_2_tipo_0 , tipo_3_tipo_0 ,
if  nofcrowns==1 && (tipo1==1 || tipo1==2 || tipo1==3) && strcmp(method,'norm')
    alpha1_in=0.2;
    alpha1_end=0.45;
    m = 0;
    p1=0.14797545390361283 ;
    p2=0.61792989644526863 ;
    p3=0.00087152935958824482;
end

if  nofcrowns==1 && (tipo1==1 || tipo1==2 || tipo1==3) && strcmp(method,'alt')
    alpha2_in=0.4;
    alpha2_end=0.8;
    m = 0;
    p1=-0.03014757086555564 ;
    p2=0.61623182089114048 ;
    p3=-0.00031796816769033765;
end

if  nofcrowns==1 && (tipo1==2) && strcmp(method,'norm')
    alpha1_in=0.1;
    alpha1_end=0.46;
    m = 0;
    p1=0.14754470386790389 ;
    p2=0.62564665417047471 ;
    p3=0.00085581515247816866;
end

if  nofcrowns==1 && (tipo1==2) && strcmp(method,'alt')
    alpha2_in=0.1;
    alpha2_end=0.75;
    m = 0;
    p1=-0.030055087249415371 ;
    p2=0.6137330796757926 ;
```

```
    p3=-0.00040030971404951229;
end

%% 2 Coronas:
if  nofcrowns==2 && (tipo1==5) && (tipo2==1 || tipo2==2 || tipo2==3) && (nofelements==4 || nofelements==6) &&
      strcmp(method,'norm')
    alpha1_in=0.10;
    alpha1_end=0.33;
    m=2.5;
    p00=0.05369592108774237 ;
    p10=1.4840260979298741  ;
    p01=-0.13862922340454287;
    p20=0.083690799575916186;
    p11=0.24034935758364004 ;
    p02=0.080901074527452668;
end

if  nofcrowns==2 && (tipo1==1) && (tipo2==5) && strcmp(method,'norm')
    alpha1_in=0.1;
    alpha1_end=0.3;
    m=2.5;
        p00=-0.0016956582466515792;
        p10=1.6293726784946525    ;
        p01=0.003781757292025012  ;
        p20=0.087131749343798678  ;
        p11=0.032872794656677438  ;
        p02=-0.0090670572714925835;

end

if  nofcrowns==2 && (tipo1==1) && (tipo2==5) && strcmp(method,'alt')
    alpha2_in=0.1;
    alpha2_end=0.49;
    m=1.75;
        p00=0.0013366475629475313 ;
        p10=0.61196340890791623   ;
        p01=-0.0023814962177391348;
        p20=-0.01741752944192038  ;
        p11=-0.011292193014912784 ;
        p02=0.0053390577732086043 ;
end

if  nofcrowns==2 && (tipo1==2) && (tipo2==4) && strcmp(method,'norm')
    alpha1_in=0.1;
    alpha1_end=0.33;
    m=2.5;
    p00=0.05998554407079424 ;
    p10=1.4878447535747987  ;
    p01=-0.16085883515471833;
    p20=0.080277794432624131;
    p11=0.23540347431674527 ;
    p02=0.10120929460481382 ;
end

if  nofcrowns==2 && (tipo1==3) && (tipo2==5) && strcmp(method,'norm')
    alpha1_in=0.1;
    alpha1_end=0.49;
    m=1.75;
    p00=0.0014240873920415501 ;
    p10=1.6215974463136227    ;
    p01=0.00043659039038128404;
    p20=0.14201310215532642   ;
    p11=0.017312570903829896  ;
    p02=-0.0042252067773094646;
end

if  nofcrowns==2 && (tipo1==3) && (tipo2==5) && strcmp(method,'alt')
    alpha2_in=0.1;
    alpha2_end=0.49;
    m=1.75;
    p00=-0.056886353995964749 ;
```

```
    p10=0.62449891952693604  ;
    p01=0.17136365344897944  ;
    p20=-0.0070913415456049156;
    p11=-0.040584387430728008 ;
    p02=-0.12014194315344627  ;
end

%% One-crown
if  nofcrowns==1 && strcmp(method,'norm'),
    for alpha1=alpha1_in :0.01: alpha1_end
        alpha2=alpha1+p1*alpha1^2 + p2*alpha1 + p3;
        singular_FE_sidelength =alpha1*hex_sidelength;
        alpha1_s=num2str((alpha1),4);
        alpha1_s= strrep (alpha1_s,'.','');
        alpha2_s=num2str((alpha2),'%2.15f');
        alpha2_s= strrep (alpha2_s,'.','');
        alpha2_s=alpha2_s(1:4);
        Files =dir([ '*','_',alpha1_s,'_',alpha2_s(1:end),'*.inp']);
        if isempty( Files )
            alpha2_s= strrep (num2str(alpha2+0.0001,'%2.15f'),'.','');
            alpha2_s=alpha2_s(1:4);
            Files =[];
            Files =dir([ '*','_',alpha1_s,'_',alpha2_s(1:end),'*.inp']);
            if isempty( Files )
                alpha2_s= strrep (num2str(alpha2-0.0001,'%2.15f'),'.','');
                alpha2_s=alpha2_s(1:4);
            end
        end
        new_entrance=[id alpha1_s '_' alpha2_s];
        new_entrance_inp=[id '_' alpha1_s '_' alpha2_s '.inp'];
        name_index=strvcat(name_index, new_entrance);
        file_index = strvcat ( file_index ,new_entrance_inp);
        singular_FE_sidelength_index =[ singular_FE_sidelength_index ;  singular_FE_sidelength ];
        alpha1_index=[alpha1_index,alpha1];
        alpha2_index=[alpha2_index,alpha2];
    end
end

%% One-crown alternative method
if  nofcrowns==1 && strcmp(method,'alt'),
    for alpha2=alpha2_in :0.01: alpha2_end
        alpha1=p1*alpha2^2 + p2*alpha2 + p3;
        singular_FE_sidelength =alpha1*hex_sidelength;
        alpha2_s=num2str((alpha2),4);
        alpha2_s= strrep (alpha2_s,'.','');
        alpha1_s=num2str((alpha1),'%2.15f');
        alpha1_s= strrep (alpha1_s,'.','');
        alpha1_s=alpha1_s(1:4);
        Files =dir([ '*','_',alpha1_s,'_',alpha2_s(1:end),'*.inp']);
        if isempty( Files )
            alpha1_s= strrep (num2str(alpha1+0.0001,'%2.15f'),'.','');
            alpha1_s=alpha1_s(1:4);
            Files =[];
            Files =dir([ '*','_',alpha1_s,'_',alpha2_s(1:end),'*.inp']);
            if isempty( Files )
                alpha1_s= strrep (num2str(alpha1-0.0001,'%2.15f'),'.','');
                alpha1_s=alpha1_s(1:4);
            end
        end
        new_entrance=[id alpha1_s '_' alpha2_s];
        new_entrance_inp=[id '_' alpha1_s '_' alpha2_s '.inp'];
        name_index=strvcat(name_index, new_entrance);
        file_index = strvcat ( file_index ,new_entrance_inp);
        singular_FE_sidelength_index =[ singular_FE_sidelength_index ;  singular_FE_sidelength ];
        alpha1_index=[alpha1_index,alpha1];
        alpha2_index=[alpha2_index,alpha2];
    end
end

%% Two-crowns
if  nofcrowns==2 && strcmp(method,'norm')
```

```
    for alpha1=alpha1_in:0.005:alpha1_end
%       for alpha2=alpha1*alpha2_in_n+alpha2_in_plus:0.05:0.85
%       for alpha3=alpha2*alpha3_in_n+alpha3_in_plus:0.05:0.85

        alpha3=m*alpha1;
        alpha2=p00 + p10*alpha1 + p01*alpha3 + p20*alpha1^2 + p11*alpha1*alpha3 + p02*alpha3^2;

        % Singular element side-length
        singular_FE_sidelength=alpha1*hex_sidelength;

        % alpha1
        alpha1_s=num2str((alpha1),4);
        alpha1_s=strrep(alpha1_s,'.','');

        % alpha2
        alpha2_s=num2str((alpha2),'%2.15f');
        alpha2_s=strrep(alpha2_s,'.','');
        alpha2_s=alpha2_s(1:4);

        % alpha3
        alpha3_s=num2str((alpha3),'%2.15f');
        alpha3_s=strrep(alpha3_s,'.','');
        if length(alpha3_s)>3, alpha3_s=alpha3_s(1:4); else, alpha3_s=alpha3_s(1:end); end
%       if alpha3_s(4)=='0', alpha3_s=alpha3_s(1:3); end
        if alpha3_s(4)=='0' && alpha3_s(3)=='0', alpha3_s(3:4)=[]; end
        if size(alpha3_s,2)==4, if alpha3_s(4)=='0', alpha3_s(4)=[]; end; end


        % = alpha2, = alpha3
        Files=dir(['*','_',alpha1_s,'_',alpha2_s(1:end),'_',alpha3_s(1:end),'*.inp']);
        if isempty(Files),
            % = alpha2, + alpha3.
            alpha2_aux=str2num([alpha2_s(1),'.',alpha2_s(2:end)]);
            alpha2_saux=strrep(num2str(alpha2_aux+0.001,'%2.15f'),'.','');
            alpha3_aux=str2num([alpha3_s(1),'.',alpha3_s(2:end)]);
            alpha3_saux=strrep(num2str(alpha3_aux+0.001,'%2.15f'),'.','');
            alpha3_saux=alpha3_saux(1:4);
            Files=dir(['*','_',alpha1_s,'_',alpha2_saux(1:end),'_',alpha3_saux(1:end),'*.inp']);
            if ~isempty(Files),
                alpha2_s=alpha2_saux;
                alpha3_s=alpha3_saux;
            else
            % = alpha2, -alpha3.
            alpha2_aux=str2num([alpha2_s(1),'.',alpha2_s(2:end)]);
            alpha3_aux=str2num([alpha3_s(1),'.',alpha3_s(2:end)]);
            alpha3_saux=strrep(num2str(alpha3_aux-0.001,'%2.15f'),'.','');
            alpha3_saux=alpha3_saux(1:4);
            Files=dir(['*','_',alpha1_s,'_',alpha2_saux(1:end),'_',alpha3_saux(1:end),'*.inp']);
            if ~isempty(Files),
                alpha2_s=alpha2_saux;
                alpha3_s=alpha3_saux;
            else
            % + alpha2, = alpha3
            alpha2_aux=str2num([alpha2_s(1),'.',alpha2_s(2:end)]);
            alpha3_aux=str2num([alpha3_s(1),'.',alpha3_s(2:end)]);
            alpha2_saux=strrep(num2str(alpha2_aux+0.001,'%2.15f'),'.','');
            alpha2_saux=alpha2_saux(1:4);
            Files=dir(['*','_',alpha1_s,'_',alpha2_saux(1:end),'_',alpha3_saux(1:end),'*.inp']);
            if ~isempty(Files),
                alpha2_s=alpha2_saux;
                alpha3_s=alpha3_saux;
            else
            % + alpha2, + alpha3
            alpha2_aux=str2num([alpha2_s(1),'.',alpha2_s(2:end)]);
            alpha3_aux=str2num([alpha3_s(1),'.',alpha3_s(2:end)]);
            alpha2_saux=strrep(num2str(alpha2_aux+0.001,'%2.15f'),'.','');
            alpha2_saux=alpha2_saux(1:4);
            alpha3_saux=strrep(num2str(alpha3_aux+0.001,'%2.15f'),'.','');
            alpha3_saux=alpha3_saux(1:4);
            Files=dir(['*','_',alpha1_s,'_',alpha2_saux(1:end),'_',alpha3_saux(1:end),'*.inp']);
            if ~isempty(Files),
```

```
                    alpha2_s=alpha2_saux;
                    alpha3_s=alpha3_saux;
                else
                % + alpha2, - alpha3
                alpha2_aux=str2num([alpha2_s(1),'.',alpha2_s(2:end)]);
                alpha3_aux=str2num([alpha3_s(1),'.',alpha3_s(2:end)]);
                alpha2_saux=strrep(num2str(alpha2_aux+0.001,'%2.15f'),'.','');
                alpha2_saux=alpha2_saux(1:4);
                alpha3_saux=strrep(num2str(alpha3_aux-0.001,'%2.15f'),'.','');
                alpha3_saux=alpha3_saux(1:4);
                Files=dir(['*','_',alpha1_s,'_',alpha2_saux(1:end),'_',alpha3_saux(1:end),'*.inp']);
                if ~isempty(Files),
                    alpha2_s=alpha2_saux;
                    alpha3_s=alpha3_saux;
                else
                % - alpha2, = alpha3
                alpha2_aux=str2num([alpha2_s(1),'.',alpha2_s(2:end)]);
                alpha3_aux=str2num([alpha3_s(1),'.',alpha3_s(2:end)]);
                alpha2_saux=strrep(num2str(alpha2_aux-0.001,'%2.15f'),'.','');
                alpha2_saux=alpha2_saux(1:4);
                Files=dir(['*','_',alpha1_s,'_',alpha2_saux(1:end),'_',alpha3_saux(1:end),'*.inp']);
                if ~isempty(Files),
                    alpha2_s=alpha2_saux;
                    alpha3_s=alpha3_saux;
                else
                % - alpha2, + alpha3
                alpha2_aux=str2num([alpha2_s(1),'.',alpha2_s(2:end)]);
                alpha3_aux=str2num([alpha3_s(1),'.',alpha3_s(2:end)]);
                alpha2_saux=strrep(num2str(alpha2_aux-0.001,'%2.15f'),'.','');
                alpha2_saux=alpha2_saux(1:4);
                alpha3_saux=strrep(num2str(alpha3_aux+0.001,'%2.15f'),'.','');
                alpha3_saux=alpha3_saux(1:4);
                Files=dir(['*','_',alpha1_s,'_',alpha2_saux(1:end),'_',alpha3_saux(1:end),'*.inp']);
                if ~isempty(Files),
                    alpha2_s=alpha2_saux;
                    alpha3_s=alpha3_saux;
                else
                % - alpha2, - alpha3
                alpha2_aux=str2num([alpha2_s(1),'.',alpha2_s(2:end)]);
                alpha3_aux=str2num([alpha3_s(1),'.',alpha3_s(2:end)]);
                alpha2_saux=strrep(num2str(alpha2_aux-0.001,'%2.15f'),'.','');
                alpha2_saux=alpha2_saux(1:4);
                alpha3_saux=strrep(num2str(alpha3_aux-0.001,'%2.15f'),'.','');
                alpha3_saux=alpha3_saux(1:4);
                Files=dir(['*','_',alpha1_s,'_',alpha2_saux(1:end),'_',alpha3_saux(1:end),'*.inp']);
                if ~isempty(Files),
                    alpha2_s=alpha2_saux;
                    alpha3_s=alpha3_saux;
                end
                end
                end
                end
                end
                end
                end
                end
            end
            if isempty(Files)
                new_entrance_inp=[];
                singular_FE_sidelength_index=[singular_FE_sidelength_index ; []];
                name_index=strvcat(name_index, new_entrance);
                alpha1_index=[alpha1_index ,[]];
                alpha2_index=[alpha2_index ,[]];
                alpha3_index=[alpha3_index ,[]];
            else
                new_entrance_inp=[id '_' alpha1_s '_' alpha2_s '_' alpha3_s '.inp'];
                name_index=strvcat(name_index, new_entrance);
                file_index = strvcat( file_index ,new_entrance_inp);
                singular_FE_sidelength_index=[singular_FE_sidelength_index ; singular_FE_sidelength ];
                alpha1_index=[alpha1_index,alpha1 ];
                alpha2_index=[alpha2_index,alpha2 ];
```

```
                alpha3_index=[alpha3_index,alpha3];
            end
        end
end

ncasos=length(file_index);

%% Two-crowns, alternative method
if nofcrowns==2 && strcmp(method,'alt')
    for alpha2=alpha2_in:0.001:alpha2_end
%         for alpha2=alpha1*alpha2_in_n+alpha2_in_plus:0.05:0.85
%         for alpha3=alpha2*alpha3_in_n+alpha3_in_plus:0.05:0.85

        alpha3=m*alpha2;
        alpha1=p00 + p10*alpha2 + p01*alpha3 + p20*alpha2^2 + p11*alpha2*alpha3 + p02*alpha3^2;

        % Singular element side-length
        singular_FE_sidelength=alpha1*hex_sidelength;

        % alpha1
        alpha1_s=num2str((alpha1),'%2.15f');
        alpha1_s=strrep(alpha1_s,'.','');
        alpha1_s=alpha1_s(1:5);

        % alpha2
        alpha2_s=num2str((alpha2),4);
        alpha2_s=strrep(alpha2_s,'.','');

        % alpha3
        alpha3_s=num2str((alpha3),'%2.15f');
        alpha3_s=strrep(alpha3_s,'.','');
        if length(alpha3_s)>3, alpha3_s=alpha3_s(1:4); else, alpha3_s=alpha3_s(1:end); end
% %         if alpha3_s(4)=='0', alpha3_s=alpha3_s(1:3); end
%         if alpha3_s(4)=='0' && alpha3_s(3)=='0', alpha3_s(3:4)=[]; end
%         if size(alpha3_s,2)==4, if alpha3_s(4)=='0', alpha3_s(4)=[]; end; end


        % = alpha2, = alpha3
        Files=dir(['*','_',alpha1_s,'_',alpha2_s(1:end),'_',alpha3_s(1:end),'*.inp']);
        if isempty(Files),
            % = alpha2, + alpha3.
            alpha2_aux=str2num([alpha2_s(1),'.',alpha2_s(2:end)]);
            alpha2_saux=strrep(num2str(alpha2_aux+0.001,'%2.15f'),'.','');
            alpha3_aux=str2num([alpha3_s(1),'.',alpha3_s(2:end)]);
            alpha3_saux=strrep(num2str(alpha3_aux+0.001,'%2.15f'),'.','');
            alpha3_saux=alpha3_saux(1:4);
            Files=dir(['*','_',alpha1_s,'_',alpha2_saux(1:end),'_',alpha3_saux(1:end),'*.inp']);
            if ~isempty(Files),
                alpha2_s=alpha2_saux;
                alpha3_s=alpha3_saux;
            else
            % = alpha2, -alpha3.
            alpha2_aux=str2num([alpha2_s(1),'.',alpha2_s(2:end)]);
            alpha3_aux=str2num([alpha3_s(1),'.',alpha3_s(2:end)]);
            alpha3_saux=strrep(num2str(alpha3_aux-0.001,'%2.15f'),'.','');
            alpha3_saux=alpha3_saux(1:4);
            Files=dir(['*','_',alpha1_s,'_',alpha2_saux(1:end),'_',alpha3_saux(1:end),'*.inp']);
            if ~isempty(Files),
                alpha2_s=alpha2_saux;
                alpha3_s=alpha3_saux;
            else
            % + alpha2, = alpha3
            alpha2_aux=str2num([alpha2_s(1),'.',alpha2_s(2:end)]);
            alpha3_aux=str2num([alpha3_s(1),'.',alpha3_s(2:end)]);
            alpha2_saux=strrep(num2str(alpha2_aux+0.001,'%2.15f'),'.','');
            alpha2_saux=alpha2_saux(1:4);
            Files=dir(['*','_',alpha1_s,'_',alpha2_saux(1:end),'_',alpha3_saux(1:end),'*.inp']);
            if ~isempty(Files),
                alpha2_s=alpha2_saux;
                alpha3_s=alpha3_saux;
            else
```

```
                % + alpha2, + alpha3
                alpha2_aux=str2num([alpha2_s(1),'.',alpha2_s(2:end)]);
                alpha3_aux=str2num([alpha3_s(1),'.',alpha3_s(2:end)]);
                alpha2_saux=strrep(num2str(alpha2_aux+0.001,'%2.15f'),'.','');
                alpha2_saux=alpha2_saux(1:4);
                alpha3_saux=strrep(num2str(alpha3_aux+0.001,'%2.15f'),'.','');
                alpha3_saux=alpha3_saux(1:4);
                Files=dir(['*','_',alpha1_s,'_',alpha2_saux(1:end),'_',alpha3_saux(1:end),'*.inp']);
                if ~isempty(Files),
                    alpha2_s=alpha2_saux;
                    alpha3_s=alpha3_saux;
                else
                % + alpha2, - alpha3
                alpha2_aux=str2num([alpha2_s(1),'.',alpha2_s(2:end)]);
                alpha3_aux=str2num([alpha3_s(1),'.',alpha3_s(2:end)]);
                alpha2_saux=strrep(num2str(alpha2_aux+0.001,'%2.15f'),'.','');
                alpha2_saux=alpha2_saux(1:4);
                alpha3_saux=strrep(num2str(alpha3_aux-0.001,'%2.15f'),'.','');
                alpha3_saux=alpha3_saux(1:4);
                Files=dir(['*','_',alpha1_s,'_',alpha2_saux(1:end),'_',alpha3_saux(1:end),'*.inp']);
                if ~isempty(Files),
                    alpha2_s=alpha2_saux;
                    alpha3_s=alpha3_saux;
                else
                % - alpha2, = alpha3
                alpha2_aux=str2num([alpha2_s(1),'.',alpha2_s(2:end)]);
                alpha3_aux=str2num([alpha3_s(1),'.',alpha3_s(2:end)]);
                alpha2_saux=strrep(num2str(alpha2_aux-0.001,'%2.15f'),'.','');
                alpha2_saux=alpha2_saux(1:4);
                Files=dir(['*','_',alpha1_s,'_',alpha2_saux(1:end),'_',alpha3_saux(1:end),'*.inp']);
                if ~isempty(Files),
                    alpha2_s=alpha2_saux;
                    alpha3_s=alpha3_saux;
                else
                % - alpha2, + alpha3
                alpha2_aux=str2num([alpha2_s(1),'.',alpha2_s(2:end)]);
                alpha3_aux=str2num([alpha3_s(1),'.',alpha3_s(2:end)]);
                alpha2_saux=strrep(num2str(alpha2_aux-0.001,'%2.15f'),'.','');
                alpha2_saux=alpha2_saux(1:4);
                alpha3_saux=strrep(num2str(alpha3_aux+0.001,'%2.15f'),'.','');
                alpha3_saux=alpha3_saux(1:4);
                Files=dir(['*','_',alpha1_s,'_',alpha2_saux(1:end),'_',alpha3_saux(1:end),'*.inp']);
                if ~isempty(Files),
                    alpha2_s=alpha2_saux;
                    alpha3_s=alpha3_saux;
                else
                % - alpha2, - alpha3
                alpha2_aux=str2num([alpha2_s(1),'.',alpha2_s(2:end)]);
                alpha3_aux=str2num([alpha3_s(1),'.',alpha3_s(2:end)]);
                alpha2_saux=strrep(num2str(alpha2_aux-0.001,'%2.15f'),'.','');
                alpha2_saux=alpha2_saux(1:4);
                alpha3_saux=strrep(num2str(alpha3_aux-0.001,'%2.15f'),'.','');
                alpha3_saux=alpha3_saux(1:4);
                Files=dir(['*','_',alpha1_s,'_',alpha2_saux(1:end),'_',alpha3_saux(1:end),'*.inp']);
                if ~isempty(Files),
                    alpha2_s=alpha2_saux;
                    alpha3_s=alpha3_saux;
                end
                end
                end
                end
                end
                end
                end
                end
            end
            if isempty(Files)
                new_entrance_inp=[];
                singular_FE_sidelength_index=[singular_FE_sidelength_index; []];
                name_index=strvcat(name_index, new_entrance);
                alpha1_index=[alpha1_index ,[]];
```

```
                alpha2_index=[alpha2_index ,[]];
                alpha3_index=[alpha3_index ,[]];
            else
                new_entrance_inp=[id '_' alpha1_s '_' alpha2_s '_' alpha3_s '.inp'];
                name_index=strvcat(name_index, new_entrance);
                 file_index = strvcat ( file_index ,new_entrance_inp);
                 singular_FE_sidelength_index =[ singular_FE_sidelength_index ;  singular_FE_sidelength ];
                alpha1_index=[alpha1_index,alpha1 ];
                alpha2_index=[alpha2_index,alpha2 ];
                alpha3_index=[alpha3_index,alpha3 ];
            end
        end
end

ncasos=length( file_index );
```

Script 9.9 Step 03 second study.

step 13

```
function [base] = step_13_numerical_regressions (alpha1_index,alpha2_index,alpha3_index, desplazamientos_calculo ,
    coordenadas_calculo ,tipo1 ,tipo2 ,id , separadores ,coordenadas, rcts ,m)
cd([ 'C:\Users\Usuario\Desktop\TFG (Avances y Bibliografia )\Macros\',id (1: separadores (1)-1), '\',id ]);
% clc
base =[];
%% X: alpha_2, Y: alpha_3, Z: u_ct; X coordinate: alpha_2; Y coordinate: alpha_3; Z coordinate: u_ct

%% Variables initialization
leyenda =[]; ajuste =[];
x_pr =[]; y_pr =[]; z_pr =[];
casos_buenos =[]; slope_index =[]; level_curve =[]; point_vector =[]; max_2nd_crown_change=[];
change=[1]; change2=[1];

%% This section fills sparce matrices (FEM results).
desplazamientos_calculo = full ( desplazamientos_calculo );
coordenadas_calculo= full ( coordenadas_calculo );

%% This section finds where the indices change, that is, all the cases with the same alpha_1 value.
for s=1:length (alpha1_index )-1
    if alpha1_index (s)-alpha1_index(s+1)~=0 % Each value that does not coincide with the following
        change=[change, s+1]; % its position is added as the first unequal values .
    end
end

for s=1:length ( desplazamientos_calculo )-1
    if desplazamientos_calculo (s)< rcts && desplazamientos_calculo(s+1)> rcts % Each value that does not coincide
      with the following
        base=[base, s ]; % its position is added as the first unequal values .
    end
end

plot (alpha1_index, desplazamientos_calculo , 'Linewidth' ,2)
% xq=[alpha1_in :0.001: alpha1_end ];
% sp=pchip(alpha1_index, desplazamientos_calculo ,xq);
% plot (xq,sp ,' Linewidth ',2)
hold on
plot (alpha1_index, rcts *ones( size (alpha1_index )));
legend( 'Simulation ', 'Correct value ', 'Location ', ' southeast ')
 title ([ 'Optimal \alpha_1 study for n= ',num2str(m)])
xlabel ( '\alpha_1 '); ylabel ( 'u_c_t ');
axis ([ min(alpha1_index) max(alpha1_index) min( desplazamientos_calculo )-0.001 max(desplazamientos_calculo )+0.001])
cd([ 'C:\Users\Usuario\Desktop\TFG (Avances y Bibliografia )\Macros 2' ]);
% print ('-dpng', '-r300 ', [' graph_estudio_2_ ', id ,' _m_',num2str(m) ,'. png ']) ;

%% This section calculates the intersection between u_ct and alpha_1. A spline is generated with the previously
    calculated points (the FEM results).
fun = @(x) 2.25596 - spline (alpha1_index, desplazamientos_calculo ,x);
x=sym('x', ' positive ');
desiredx1 = fzero (fun, 0.24);
```

```
desiredx2 = fzero(fun,0.3);
[desiredx1 desiredx2 max(desplazamientos_calculo) min(desplazamientos_calculo)]

cd([ 'C:\Users\Usuario\Desktop\TFG (Avances y Bibliografia )\Macros\Codigo Matlab']);
```

Script 9.10 Step 13 second study.

9.1.5 Python Abaqus Macros Scripts

Only some cases are presented here, the remaining ones can be found by changing the different cuts obtained by the aforementioned MATLAB scripts, and consequently changing the regions and lines to which we apply the following steps. In total there is one of each of these Macros for every chosen configuration in each study. In total, 300 for the first study, 12 for the second study, and 39 for the third study.

Study 1

```
def macro_8E_tipo_1_tipo_1():
    tipo1=1
    tipo2=1
#_____________________________________________________________ Cálculo de vértices de agujero y corona
    nofelements=8 # Introducir número de elementos
#
#     for alpha1 in wp.arange (0.05,0.75,0.05) :
#         alpha2min=alpha1*sqrt(2)
#         for alpha2 in wp.arange(alpha2min,0.75,0.05) : # el segundo es 0.75
    sidelength=0.15
    xpaux=sidelength*wp.array ([1,0.5,0,-0.5,-1])
    ypaux=sidelength*wp.array([0,0.866025403784439,0.866025403784439,0.866025403784439,0])
#     for alpha1 in wp.arange (0.05,0.75,0.05) :
#         alpha2min=alpha1*sqrt(2)
#         for alpha2 in wp.arange(alpha2min,0.75,0.05) : # el segundo es 0.75
    for alpha1 in wp.arange (0.05,0.85,0.05) :
        alpha2min=alpha1*1/(cos(pi/nofelements))
        for alpha2 in wp.arange(alpha2min,0.85,0.05) : # el segundo es 0.75
            alpha3min=alpha2*1/(cos(pi/nofelements))
            for alpha3 in wp.arange(alpha3min,0.85,0.05) : # el segundo es 0.75
                N = 2*nofelements
                n = list(range(0, nofelements+1))
                n2= list(range(0, nofelements+1))
                n3=list(range(0, nofelements+1))
                rci= sidelength*alpha1
                rco= sidelength*alpha2
                rco2=sidelength*alpha3
                x1= list(range(0, nofelements+1))
                y1= list(range(0, nofelements+1))
                x2= list(range(0, nofelements+1))
                y2= list(range(0, nofelements+1))
                x3= list(range(0, nofelements+1))
                y3= list(range(0, nofelements+1))
                for i in range(0,nofelements+1):
                    x1[i]=rci*cos(2*pi*n[i]/N)
                    y1[i]=rci*sin(2*pi*n[i]/N)
                for i in range(0,nofelements+1):
                    x2[i]=rco*cos(2*pi*n2[i]/(N))
                    y2[i]=rco*sin(2*pi*n2[i]/(N))
                for i in range(0,nofelements+1):
                    x3[i]=rco2*cos(2*pi*n3[i]/(N))
                    y3[i]=rco2*sin(2*pi*n3[i]/(N))
#_____________________________________________________________ Definición xcorte e ycorte

                xcorte = list(range(0, (nofelements*2)+1))
                ycorte = list(range(0, (nofelements*2)+1))
                idx = 0
                mitad=nofelements/2
                for i in range(0,nofelements+1):
                    xcorte[idx]=x2[i]
                    ycorte[idx]=y2[i]
                    idx=idx+1
```

```
                    if i < mitad :
                        xcorte[idx]=x1[i+1]
                        ycorte[idx]=y1[i+1]
                    else :
                        if i < nofelements:
                            xcorte[idx]=x1[i]
                            ycorte[idx]=y1[i]
                    idx=idx+1
#_____________________________________# p01
                Mdb()
                session.viewports['Viewport: 1'].setValues(displayedObject=None)
                s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=2.0)
                g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
                s.setPrimaryObject(option=STANDALONE)
                s.rectangle(point1=(-0.5, 0.0), point2=(0.5, 1.0))
                p = mdb.models['Model-1'].Part(name='Part-1', dimensionality=TWO_D_PLANAR,
                    type=DEFORMABLE_BODY)
                p = mdb.models['Model-1'].parts['Part-1']
                p.BaseShell(sketch=s)
                s.unsetPrimaryObject()
                p = mdb.models['Model-1'].parts['Part-1']
                session.viewports['Viewport: 1'].setValues(displayedObject=p)
                del mdb.models['Model-1'].sketches['__profile__']
#_____________________________________# p02
                p = mdb.models['Model-1'].parts['Part-1']
                f1, e1, d2 = p.faces, p.edges, p.datums
                t = p.MakeSketchTransform(sketchPlane=f1[0], sketchPlaneSide=SIDE1, origin=(
                    0.0, 0.5, 0.0))
                s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=2.82,
                    gridSpacing=0.07, transform=t)
                g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
                s.setPrimaryObject(option=SUPERIMPOSE)
                p = mdb.models['Model-1'].parts['Part-1']
                p.projectReferencesOntoSketch(sketch=s, filter=COPLANAR_EDGES)
                s.Line(point1=(0.0, -0.51), point2=(0.0+xpaux[0], -0.5+ypaux[0]))
                s.Line(point1=(0.0+xpaux[0], -0.5+ypaux[0]), point2=(0.0+xpaux[1], -0.5+ypaux[1]))
                s.Line(point1=(0.0+xpaux[1], -0.5+ypaux[1]), point2=(0.0+xpaux[2], -0.5+ypaux[2]))
                s.Line(point1=(0.0+xpaux[2], -0.5+ypaux[2]), point2=(0.0+xpaux[3], -0.5+ypaux[3]))
                s.Line(point1=(0.0+xpaux[3], -0.5+ypaux[3]), point2=(0.0+xpaux[4], -0.5+ypaux[4]))
                s.Line(point1=(0.0+xpaux[4], -0.5+ypaux[4]), point2=(0.0, -0.51))
                s.Line(point1=(0.5, 0.5), point2=(0.0+xpaux[1], -0.5+ypaux[1]))
                s.Line(point1=(-0.5, 0.5), point2=(0.0+xpaux[3], -0.5+ypaux[3]))
                s.Line(point1=(0.0, -0.5), point2=(0.0, 0.5))
                p = mdb.models['Model-1'].parts['Part-1']
                f = p.faces
                pickedFaces = f.getSequenceFromMask(mask=('[#1 ]', ), )
                e, d1 = p.edges, p.datums
                p.PartitionFaceBySketch(faces=pickedFaces, sketch=s)
                s.unsetPrimaryObject()
                del mdb.models['Model-1'].sketches['__profile__']
#_____________________________________# p03
                p = mdb.models['Model-1'].parts['Part-1']
                f, e1, d2 = p.faces, p.edges, p.datums
                t = p.MakeSketchTransform(sketchPlane=f[1], sketchPlaneSide=SIDE1, origin=(
                    0.097222, 0.096225, 0.0))
                s1 = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
                    sheetSize=1.11, gridSpacing=0.02, transform=t)
                g, v, d, c = s1.geometry, s1.vertices, s1.dimensions, s1.constraints
                s1.setPrimaryObject(option=SUPERIMPOSE)
                p = mdb.models['Model-1'].parts['Part-1']
                p.projectReferencesOntoSketch(sketch=s1, filter=COPLANAR_EDGES)
                session.viewports['Viewport: 1'].view.setValues(nearPlane=0.525633,
                    farPlane=0.797243, width=0.993297, height=0.498788, cameraPosition=(
                    0.123004, 0.205139, 0.661438), cameraTarget=(0.123004, 0.205139, 0))
                s1.sketchOptions.setValues(gridOrigin=(-0.097222, -0.096225))
                s1.CircleByCenterPerimeter(center=(-0.097222, -0.096225), point1=(-0.097222+rci, -0.096225))
                s1.CircleByCenterPerimeter(center=(-0.097222, -0.096225), point1=(-0.097222+rco, -0.096225))
                p = mdb.models['Model-1'].parts['Part-1']
                f = p.faces
                pickedFaces = f.getSequenceFromMask(mask=('[#a ]', ), )
                e, d1 = p.edges, p.datums
```

```
                    p.PartitionFaceBySketch(faces=pickedFaces, sketch=s1)
                    s1.unsetPrimaryObject()
                    del mdb.models['Model-1'].sketches['__profile__']
#_____________________________________# p04
                    session.viewports['Viewport: 1'].view.setValues(nearPlane=2.63361,
                        farPlane=3.02325, width=1.10348, height=0.554118, cameraPosition=(
                        0.0496831, 0.207573, 2.82843), cameraTarget=(0.0496831, 0.207573, 0))
                    p = mdb.models['Model-1'].parts['Part-1']
                    f1 = p.faces
                    p.RemoveFaces(faceList = f1[5:6]+f1[7:8], deleteCells=False)
#_____________________________________# p05
                    p = mdb.models['Model-1'].parts['Part-1']
                    f, e1, d2 = p.faces, p.edges, p.datums
                    t = p.MakeSketchTransform(sketchPlane=f[1], sketchPlaneSide=SIDE1, origin=(
                        -0.082041, 0.082041, 0.0))
                    s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=0.91,
                        gridSpacing=0.02, transform=t)
                    g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
                    s.setPrimaryObject(option=SUPERIMPOSE)
                    p = mdb.models['Model-1'].parts['Part-1']
                    p.projectReferencesOntoSketch(sketch=s, filter=COPLANAR_EDGES)
                    session.viewports['Viewport: 1'].view.setValues(nearPlane=0.351973,
                        farPlane=0.518738, width=0.609871, height=0.306249, cameraPosition=(
                        0.0559457, 0.0413319, 0.435355), cameraTarget=(0.0559457, 0.0413319,
                        -3.46945e-18))
                    s.Line(point1=(0.082041+x2[0],-0.082041+y2[0]), point2=(0.082041+x1[1],-0.082041+y1[1]))
                    s.Line(point1=(0.082041+x1[1],-0.082041+y1[1]), point2=(0.082041+x2[1],-0.082041+y2[1]))
                    s.Line(point1=(0.082041+x2[1],-0.082041+y2[1]), point2=(0.082041+x1[2],-0.082041+y1[2]))
                    s.Line(point1=(0.082041+x1[2],-0.082041+y1[2]), point2=(0.082041+x2[2],-0.082041+y2[2]))
                    s.Line(point1=(0.082041+x2[2],-0.082041+y2[2]), point2=(0.082041+x1[3],-0.082041+y1[3]))
                    s.Line(point1=(0.082041+x1[3],-0.082041+y1[3]), point2=(0.082041+x2[3],-0.082041+y2[3]))
                    s.Line(point1=(0.082041+x2[3],-0.082041+y2[3]), point2=(0.082041+x1[4],-0.082041+y1[4]))
                    s.Line(point1=(0.082041+x1[4],-0.082041+y1[4]), point2=(0.082041+x2[4],-0.082041+y2[4]))
                    s.Line(point1=(0.082041+x1[4],-0.082041+y1[4]), point2=(0.082041+x2[5],-0.082041+y2[5]))
                    s.Line(point1=(0.082041+x2[5],-0.082041+y2[5]), point2=(0.082041+x1[5],-0.082041+y1[5]))
                    s.Line(point1=(0.082041+x1[5],-0.082041+y1[5]), point2=(0.082041+x2[6],-0.082041+y2[6]))
                    s.Line(point1=(0.082041+x2[6],-0.082041+y2[6]), point2=(0.082041+x1[6],-0.082041+y1[6]))
                    s.Line(point1=(0.082041+x1[6],-0.082041+y1[6]), point2=(0.082041+x2[7],-0.082041+y2[7]))
                    s.Line(point1=(0.082041+x2[7],-0.082041+y2[7]), point2=(0.082041+x1[7],-0.082041+y1[7]))
                    s.Line(point1=(0.082041+x1[7],-0.082041+y1[7]), point2=(0.082041+x2[8],-0.082041+y2[8]))
                    p = mdb.models['Model-1'].parts['Part-1']
                    f = p.faces
                    pickedFaces = f.getSequenceFromMask(mask=('[#a ]', ), )
                    e, d1 = p.edges, p.datums
                    p.PartitionFaceBySketch(faces=pickedFaces, sketch=s)
                    s.unsetPrimaryObject()
                    del mdb.models['Model-1'].sketches['__profile__']
#_____________________________________# p06
                    p = mdb.models['Model-1'].parts['Part-1']
                    f, e, d = p.faces, p.edges, p.datums
                    t = p.MakeSketchTransform(sketchPlane=f[14], sketchPlaneSide=SIDE1, origin=(
                        -0.065649, 0.064922, 0.0))
                    s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
                        sheetSize=0.396, gridSpacing=0.009, transform=t)
                    g, v, d1, c = s.geometry, s.vertices, s.dimensions, s.constraints
                    s.sketchOptions.setValues(decimalPlaces=3)
                    s.setPrimaryObject(option=SUPERIMPOSE)
                    p = mdb.models['Model-1'].parts['Part-1']
                    p.projectReferencesOntoSketch(sketch=s, filter=COPLANAR_EDGES)
                    session.viewports['Viewport: 1'].view.setValues(nearPlane=0.304268,
                        farPlane=0.489457, width=0.677248, height=0.340082, cameraPosition=(
                        0.0752357, 0.0966471, 0.396863), cameraTarget=(0.0752357, 0.0966471,
                        0))
                    s.sketchOptions.setValues(gridOrigin=(0.065649, -0.064922))
                    s.CircleByCenterPerimeter(center=(0.065649, -0.064922), point1=(0.065649+rco2, -0.064922))
                    s.CoincidentConstraint(entity1=v[29], entity2=g[48], addUndoState=False)
                    p = mdb.models['Model-1'].parts['Part-1']
                    f = p.faces
                    pickedFaces = f.getSequenceFromMask(mask=('[#14000 ]', ), )
                    e1, d2 = p.edges, p.datums
                    p.PartitionFaceBySketch(faces=pickedFaces, sketch=s)
```

```
                    s.unsetPrimaryObject()
                    del mdb.models['Model-1'].sketches['__profile__']
#_____________________________________# p07
                    p = mdb.models['Model-1'].parts['Part-1']
                    f1, e, d = p.faces, p.edges, p.datums
                    t = p.MakeSketchTransform(sketchPlane=f1[0], sketchPlaneSide=SIDE1, origin=(
                        0.047228, 0.047228, 0.0))
                    s1 = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
                        sheetSize=0.54, gridSpacing=0.01, transform=t)
                    g, v, d1, c = s1.geometry, s1.vertices, s1.dimensions, s1.constraints
                    s1.sketchOptions.setValues(decimalPlaces=3)
                    s1.setPrimaryObject(option=SUPERIMPOSE)
                    p = mdb.models['Model-1'].parts['Part-1']
                    p.projectReferencesOntoSketch(sketch=s1, filter=COPLANAR_EDGES)
                    session.viewports['Viewport: 1'].view.setValues(nearPlane=0.188861,
                        farPlane=0.312977, width=0.453902, height=0.227929, cameraPosition=(
                        0.0935676, 0.0858863, 0.250919), cameraTarget=(0.0935676, 0.0858863,
                        -3.46945e-18))
                    s1.sketchOptions.setValues(gridOrigin=(-0.047228, -0.047228))
                    s1.Line(point1=(-0.047228+x3[0],-0.047228+y3[0]), point2=(-0.047228+x2[1],-0.047228+y2[1]))
                    s1.Line(point1=(-0.047228+x2[1],-0.047228+y2[1]), point2=(-0.047228+x3[1],-0.047228+y3[1]))
                    s1.Line(point1=(-0.047228+x3[1],-0.047228+y3[1]), point2=(-0.047228+x2[2],-0.047228+y2[2]))
                    s1.Line(point1=(-0.047228+x2[2],-0.047228+y2[2]), point2=(-0.047228+x3[2],-0.047228+y3[2]))
                    s1.Line(point1=(-0.047228+x3[2],-0.047228+y3[2]), point2=(-0.047228+x2[3],-0.047228+y3[3]))
                    s1.Line(point1=(-0.047228+x2[3],-0.047228+y2[3]), point2=(-0.047228+x3[3],-0.047228+y3[3]))
                    s1.Line(point1=(-0.047228+x3[3],-0.047228+y3[3]), point2=(-0.047228+x2[4],-0.047228+y2[4]))
                    s1.Line(point1=(-0.047228+x2[4],-0.047228+y2[4]), point2=(-0.047228+x3[4],-0.047228+y3[4]))
                    s1.Line(point1=(-0.047228+x2[4],-0.047228+y2[4]), point2=(-0.047228+x3[5],-0.047228+y3[5]))
                    s1.Line(point1=(-0.047228+x3[5],-0.047228+y3[5]), point2=(-0.047228+x2[5],-0.047228+y2[5]))
                    s1.Line(point1=(-0.047228+x2[5],-0.047228+y2[5]), point2=(-0.047228+x3[6],-0.047228+y3[6]))
                    s1.Line(point1=(-0.047228+x3[6],-0.047228+y3[6]), point2=(-0.047228+x2[6],-0.047228+y2[6]))
                    s1.Line(point1=(-0.047228+x2[6],-0.047228+y2[6]), point2=(-0.047228+x3[7],-0.047228+y3[7]))
                    s1.Line(point1=(-0.047228+x3[7],-0.047228+y3[7]), point2=(-0.047228+x2[7],-0.047228+y2[7]))
                    s1.Line(point1=(-0.047228+x2[7],-0.047228+y2[7]), point2=(-0.047228+x3[8],-0.047228+y3[8]))
                    p = mdb.models['Model-1'].parts['Part-1']
                    f = p.faces
                    pickedFaces = f.getSequenceFromMask(mask=('[#10001 ]', ), )
                    e1, d2 = p.edges, p.datums
                    p.PartitionFaceBySketch(faces=pickedFaces, sketch=s1)
                    s1.unsetPrimaryObject()
                    del mdb.models['Model-1'].sketches['__profile__']
#_____________________________________# p08
                    session.viewports['Viewport: 1'].partDisplay.setValues(mesh=ON)
                    session.viewports['Viewport: 1'].partDisplay.meshOptions.setValues(
                        meshTechnique=ON)
                    session.viewports['Viewport: 1'].partDisplay.geometryOptions.setValues(
                        referenceRepresentation=OFF)
                    p = mdb.models['Model-1'].parts['Part-1']
                    e = p.edges
                    pickedEdges = e.getSequenceFromMask(mask=('[# fffffff  #3 fffffe1  ]', ), )
                    p.seedEdgeByNumber(edges=pickedEdges, number=1, constraint=FIXED)
                    p = mdb.models['Model-1'].parts['Part-1']
                    f = p.faces
                    pickedRegions = f.getSequenceFromMask(mask=('[# fffffff  #3f ]', ), )
                    p.setMeshControls(regions=pickedRegions, elemShape=TRI)
                    p = mdb.models['Model-1'].parts['Part-1']
                    p.seedPart(size=0.05, deviationFactor=0.1, minSizeFactor=0.1)
                    p = mdb.models['Model-1'].parts['Part-1']
                    f = p.faces
                    pickedRegions = f.getSequenceFromMask(mask=('[# fffffff  #3f ]', ), )
                    p.generateMesh(regions=pickedRegions)
#_____________________________________# p09
                    p = mdb.models['Model-1'].parts['Part-1']
                    p.generateMesh()
                    a = mdb.models['Model-1'].rootAssembly
                    session.viewports['Viewport: 1'].setValues(displayedObject=a)
                    session.viewports['Viewport: 1'].assemblyDisplay.setValues(
                        optimizationTasks=OFF, geometricRestrictions=OFF, stopConditions=OFF)
                    a = mdb.models['Model-1'].rootAssembly
                    a.DatumCsysByDefault(CARTESIAN)
                    p = mdb.models['Model-1'].parts['Part-1']
```

```
                    a.Instance(name='Part-1-1', part=p, dependent=ON)
                    alpha1_s=str(alpha1).replace('.', '')
                    alpha2_s=str(alpha2).replace('.', '')
                    alpha3_s=str(alpha3).replace('.', '')
                    tipo1_s=str(tipo1)
                    tipo2_s=str(tipo2)
                    nofelements_s=str(nofelements)
                    nombre=nofelements_s+"E_tipo_"+tipo1_s+"_tipo_"+tipo2_s+"_"+alpha1_s[0:5]+"_"+alpha2_s[0:4]+"_"+
    alpha3_s[0:4]
                    mdb.Job(name=nombre, model='Model-1', description='', type=ANALYSIS,
                        atTime=None, waitMinutes=0, waitHours=0, queue=None, memory=90,
                        memoryUnits=PERCENTAGE, getMemoryFromAnalysis=True,
                         explicitPrecision =DOUBLE_PLUS_PACK, nodalOutputPrecision=FULL,
                        echoPrint=OFF, modelPrint=OFF, contactPrint=OFF, historyPrint=OFF,
                        userSubroutine='', scratch='', resultsFormat=ODB)
                    mdb.jobs[nombre].writeInput(consistencyChecking=OFF)
                    mdb.saveAs(
                        pathName='C:/Users/Usuario/Desktop/TFG (Avances y Bibliografia)/Macros/'+nofelements_s+'E/'+
    nombre)
```

Script 9.11 Study 1 Abaqus Macro.

Study 2

Here the cuts are all made in the same step to avoid re-defining regions.

```
def Macro_4E_tipo_1_tipo_5b():
    tipo1=1
    tipo2=5
#_____________________________________________________ Cálculo de vértices de agujero y corona
    nofelements=4 # Introducir número de elementos
#
#    for alpha1 in wp.arange (0.05,0.75,0.05) :
#        alpha2min=alpha1*sqrt(2)
#        for alpha2 in wp.arange(alpha2min ,0.75,0.05) : # el segundo es 0.75
     sidelength=0.1
    xpaux=sidelength*wp.array ([1,0.5,0,-0.5,-1])
    ypaux=sidelength*wp.array([0,0.866025403784439,0.866025403784439,0.866025403784439,0])
    for alpha1 in wp.arange (0.1,0.3301,0.01) :
        m=2.5
        alpha3=m*alpha1
        p00=-0.0016956582466515792
        p10=1.6293726784946525
        p01=0.003781757292025012
        p20=0.087131749343798678
        p11=0.032872794656677438
        p02=-0.0090670572714925835
        alpha2=p00 + p10*alpha1 + p01*alpha3 + p20*alpha1**2 + p11*alpha1*alpha3 + p02*alpha3**2
        print [alpha1]
        print [alpha2]
        print [alpha3]
        N   = 2*nofelements
        n   = list (range(0, nofelements+1))
        n2= list (range(0, 2*nofelements+1))
        n3= list (range(0, 2*nofelements+1))
        rci= sidelength*alpha1
        rco= sidelength*alpha2
        rco2=sidelength*alpha3
        x1= list (range(0, nofelements+1))
        y1= list (range(0, nofelements+1))
        x2= list (range(0, 2*nofelements+1))
        y2= list (range(0, 2*nofelements+1))
        x3= list (range(0, 2*nofelements+1))
        y3= list (range(0, 2*nofelements+1))
        for i in range(0,nofelements+1):
            x1[i]=rci*cos(2*pi*n[i]/N)
            y1[i]=rci*sin(2*pi*n[i]/N)
        for i in range(0,2*nofelements+1):
            x2[i]=rco*cos(2*pi*n2[i]/(2*N))
            y2[i]=rco*sin(2*pi*n2[i]/(2*N))
```

```
45          for i in range(0,2*nofelements+1):
46              x3[i]=rco2*cos(2*pi*n3[i]/(2*N))
47              y3[i]=rco2*sin(2*pi*n3[i]/(2*N))
48  #_____________________________________________________________ Definición xcorte e ycorte
49          xcorte    = list(range(0, (nofelements*2)+1))
50          ycorte    = list(range(0, (nofelements*2)+1))
51          idx = 0
52          mitad=nofelements/2
53          for i in range(0, nofelements+1):
54              xcorte[idx]=x2[i]
55              ycorte[idx]=y2[i]
56              idx=idx+1
57              if i < mitad :
58                  xcorte[idx]=x1[i+1]
59                  ycorte[idx]=y1[i+1]
60              else :
61                  if i < nofelements:
62                      xcorte[idx]=x1[i]
63                      ycorte[idx]=y1[i]
64              idx=idx+1
65  #_____________________________________# p01
66          Mdb()
67          s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=2.0)
68          g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
69          s.setPrimaryObject(option=STANDALONE)
70          s.rectangle(point1=(-0.5, 0.0), point2=(0.5, 1.0))
71          p = mdb.models['Model-1'].Part(name='Part-1', dimensionality=TWO_D_PLANAR,
72              type=DEFORMABLE_BODY)
73          p = mdb.models['Model-1'].parts['Part-1']
74          p.BaseShell(sketch=s)
75          s.unsetPrimaryObject()
76          p = mdb.models['Model-1'].parts['Part-1']
77          del mdb.models['Model-1'].sketches['__profile__']
78  #_____________________________________# p02
79          p = mdb.models['Model-1'].parts['Part-1']
80          f1, e1, d2 = p.faces, p.edges, p.datums
81          t = p.MakeSketchTransform(sketchPlane=f1[0], sketchPlaneSide=SIDE1, origin=(
82              0.0, 0.0, 0.0))
83          s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=2.82,
84              gridSpacing=0.07, transform=t)
85          g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
86          s.setPrimaryObject(option=SUPERIMPOSE)
87          p = mdb.models['Model-1'].parts['Part-1']
88          p.projectReferencesOntoSketch(sketch=s, filter=COPLANAR_EDGES)
89          s.Line(point1=(0.0, -0.01), point2=(0.0+xpaux[0], ypaux[0]))
90          s.Line(point1=(0.0+xpaux[0], ypaux[0]), point2=(0.0+xpaux[1], ypaux[1]))
91          s.Line(point1=(0.0+xpaux[1], ypaux[1]), point2=(0.0+xpaux[2], ypaux[2]))
92          s.Line(point1=(0.0+xpaux[2], ypaux[2]), point2=(0.0+xpaux[3], ypaux[3]))
93          s.Line(point1=(0.0+xpaux[3], ypaux[3]), point2=(0.0+xpaux[4], ypaux[4]))
94          s.Line(point1=(0.0+xpaux[4], ypaux[4]), point2=(0.0, -0.01))
95          s.Line(point1=(0.5, 1), point2=(0.0+xpaux[1], ypaux[1]))
96          s.Line(point1=(-0.5, 1), point2=(0.0+xpaux[3], ypaux[3]))
97          s.Line(point1=(0.0, 0), point2=(0.0, 1))
98          s.CircleByCenterPerimeter(center=(0,0), point1=(rci, 0))
99          s.Line(point1=(x2[0] ,y2[0] ), point2=( x2[1], y2[1]))
100         s.Line(point1=(x2[1] ,y2[1] ), point2=( x2[3], y2[3]))
101         s.Line(point1=(x2[3] ,y2[3] ), point2=( x2[5], y2[5]))
102         s.Line(point1=(x2[5] ,y2[5] ), point2=( x2[7], y2[7]))
103         s.Line(point1=(x2[7] ,y2[7] ), point2=( x2[8], y2[8]))
104         s.Line(point1=(x1[0] ,y1[0] ), point2=( x2[1], y2[1]))
105         s.Line(point1=(x2[1] ,y2[1] ), point2=( x1[1], y1[1]))
106         s.Line(point1=(x1[1] ,y1[1] ), point2=( x2[3], y2[3]))
107         s.Line(point1=(x2[3] ,y2[3] ), point2=( x1[2], y1[2]))
108         s.Line(point1=(x1[2] ,y1[2] ), point2=( x2[5], y2[5]))
109         s.Line(point1=(x2[5] ,y2[5] ), point2=( x1[3], y1[3]))
110         s.Line(point1=(x1[3] ,y1[3] ), point2=( x2[7], y2[7]))
111         s.Line(point1=(x2[7] ,y2[7] ), point2=( x1[4], y1[4]))
112         s.CircleByCenterPerimeter(center=(0,0), point1=(rco2, 0))
113         s.Line(point1=(x3[0] ,y3[0] ), point2=( x2[1], y2[1]))
114         s.Line(point1=(x2[1] ,y2[1] ), point2=( x3[1], y3[1]))
115         s.Line(point1=(x3[1] ,y3[1] ), point2=( x2[3], y2[3]))
```

```
            s.Line(point1=(x2[3] ,y2[3] ), point2=( x3[3], y3[3]))
            s.Line(point1=(x3[3] ,y3[3] ), point2=( x2[4], y2[3]))
            s.Line(point1=(x2[4] ,y2[3] ), point2=( x3[5], y3[5]))
            s.Line(point1=(x3[5] ,y3[5] ), point2=( x2[5], y2[5]))
            s.Line(point1=(x2[5] ,y2[5] ), point2=( x3[7], y3[7]))
            s.Line(point1=(x3[7] ,y3[7] ), point2=( x2[7], y2[7]))
            s.Line(point1=(x2[7] ,y2[7] ), point2=( x3[8], y3[8]))
            p = mdb.models['Model-1'].parts['Part-1']
            f = p.faces
            pickedFaces = f.getSequenceFromMask(mask=('[#1 ]', ), )
            e, d1 = p.edges, p.datums
            p.PartitionFaceBySketch(faces=pickedFaces, sketch=s)
            s.unsetPrimaryObject()
            del mdb.models['Model-1'].sketches['__profile__']

            session.viewports['Viewport: 1'].view.setValues(nearPlane=2.75735,
                farPlane=2.8995, width=0.435438, height=0.194272,
                viewOffsetX=0.0172598, viewOffsetY=-0.4128)
            session.viewports['Viewport: 1'].partDisplay.meshOptions.setValues(
                meshTechnique=OFF)
            session.viewports['Viewport: 1'].partDisplay.geometryOptions.setValues(
                referenceRepresentation=ON)
            p = mdb.models['Model-1'].parts['Part-1']
            f = p.faces
            p.RemoveFaces(faceList = f[2:3]+f[6:7], deleteCells=False)
            session.viewports['Viewport: 1'].partDisplay.meshOptions.setValues(
                meshTechnique=ON)
            session.viewports['Viewport: 1'].partDisplay.geometryOptions.setValues(
                referenceRepresentation=OFF)
            p = mdb.models['Model-1'].parts['Part-1']
            e = p.edges
            pickedEdges = e.getSequenceFromMask(mask=('[#c3ffffff #7807e3 ]', ), )
            p.seedEdgeByNumber(edges=pickedEdges, number=1, constraint=FIXED)
            p = mdb.models['Model-1'].parts['Part-1']
            e = p.edges
            pickedEdges = e.getSequenceFromMask(mask=('[#20000000 #10000 ]', ), )
            p.seedEdgeByNumber(edges=pickedEdges, number=3)
            p = mdb.models['Model-1'].parts['Part-1']
            e = p.edges
            pickedEdges = e.getSequenceFromMask(mask=('[#0 #2010 ]', ), )
            p.seedEdgeByNumber(edges=pickedEdges, number=int(math.ceil((1-alpha3)*3/2)))
            p = mdb.models['Model-1'].parts['Part-1']
            e = p.edges
            pickedEdges = e.getSequenceFromMask(mask=('[#0 #801800 ]', ), )
            p.seedEdgeByNumber(edges=pickedEdges, number=int(math.ceil((1-alpha3)*3)))
            p = mdb.models['Model-1'].parts['Part-1']
            p.seedPart(size=0.05, deviationFactor=0.1, minSizeFactor=0.1)
            p = mdb.models['Model-1'].parts['Part-1']
            f = p.faces
            pickedRegions = f.getSequenceFromMask(mask=('[#ffffff ]', ), )
            p.setMeshControls(regions=pickedRegions, elemShape=TRI)
            p = mdb.models['Model-1'].parts['Part-1']
            p.generateMesh()

#_____________________________________# p09
            p = mdb.models['Model-1'].parts['Part-1']
            p.generateMesh()
            a = mdb.models['Model-1'].rootAssembly
            session.viewports['Viewport: 1'].setValues(displayedObject=a)
            session.viewports['Viewport: 1'].assemblyDisplay.setValues(
                optimizationTasks=OFF, geometricRestrictions=OFF, stopConditions=OFF)
            a = mdb.models['Model-1'].rootAssembly
            a.DatumCsysByDefault(CARTESIAN)
            p = mdb.models['Model-1'].parts['Part-1']
            a.Instance(name='Part-1-1', part=p, dependent=ON)
            alpha1_s=str(alpha1).replace('.', '')
            alpha2_s=str(alpha2).replace('.', '')
            alpha3_s=str(alpha3).replace('.', '')
            tipo1_s=str(tipo1)
            tipo2_s=str(tipo2)
            nofelements_s=str(nofelements)
```

```
            nombre=nofelements_s+"E_tipo_"+tipo1_s+"_tipo_"+tipo2_s+"_"+alpha1_s[0:5]+"_"+alpha2_s[0:4]+"_"+alpha3_s
        [0:4]
            mdb.Job(name=nombre, model='Model-1', description='', type=ANALYSIS,
                atTime=None, waitMinutes=0, waitHours=0, queue=None, memory=90,
                memoryUnits=PERCENTAGE, getMemoryFromAnalysis=True,
                 explicitPrecision =DOUBLE_PLUS_PACK, nodalOutputPrecision=FULL,
                echoPrint=OFF, modelPrint=OFF, contactPrint=OFF, historyPrint =OFF,
                userSubroutine='', scratch='', resultsFormat=ODB)
            mdb.jobs[nombre].writeInput(consistencyChecking=OFF)
            mdb.saveAs(
            pathName='C:/Users/Usuario/Desktop/TFG (Avances y Bibliografia )/Macros 2/'+nombre)

def Macro_4E_tipo_1_tipo_5b_str():
    tipo1=1
    tipo2=5
#_____________________________________________________________ Cálculo de vértices de agujero y corona
    nofelements=4 # Introducir número de elementos
#
#     for alpha1 in wp.arange (0.05,0.75,0.05) :
#         alpha2min=alpha1*sqrt(2)
#         for alpha2 in wp.arange(alpha2min ,0.75,0.05) : # el segundo es 0.75
     sidelength =0.1
    xpaux=sidelength*wp.array ([1,0.5,0,-0.5,-1])
    ypaux=sidelength*wp.array([0,0.866025403784439,0.866025403784439,0.866025403784439,0])
    for alpha1 in wp.arange (0.1,0.3801,0.01) :
        m=3
        alpha3=m*alpha1
        p00=-0.0016956582466515792
        p10=1.6293726784946525
        p01=0.003781757292025012
        p20=0.087131749343798678
        p11=0.032872794656677438
        p02=-0.0090670572714925835
        alpha2=p00 + p10*alpha1 + p01*alpha3 + p20*alpha1**2 + p11*alpha1*alpha3 + p02*alpha3**2
        print [alpha1]
        print [alpha2]
        print [alpha3]
        N   = 2*nofelements
        n   = list (range(0, nofelements+1))
        n2= list (range(0, 2*nofelements+1))
        n3= list(range(0, 2*nofelements+1))
        rci= sidelength*alpha1
        rco= sidelength*alpha2
        rco2= sidelength*alpha3
        x1= list (range(0, nofelements+1))
        y1= list (range(0, nofelements+1))
        x2= list (range(0, 2*nofelements+1))
        y2= list (range(0, 2*nofelements+1))
        x3= list (range(0, 2*nofelements+1))
        y3= list (range(0, 2*nofelements+1))
        for i in range(0,nofelements+1):
            x1[i]=rci*cos(2*pi*n[i]/N)
            y1[i]=rci*sin(2*pi*n[i]/N)
        for i in range(0,2*nofelements+1):
            x2[i]=rco*cos(2*pi*n2[i]/(2*N))
            y2[i]=rco*sin(2*pi*n2[i]/(2*N))
        for i in range(0,2*nofelements+1):
            x3[i]=rco2*cos(2*pi*n3[i]/(2*N))
            y3[i]=rco2*sin(2*pi*n3[i]/(2*N))
#_____________________________________________________________ Definición xcorte e ycorte
        xcorte   = list (range(0, (nofelements*2)+1))
        ycorte   = list (range(0, (nofelements*2)+1))
        idx = 0
        mitad=nofelements/2
        for i in range(0,nofelements+1):
            xcorte[idx]=x2[i]
            ycorte[idx]=y2[i]
            idx=idx+1
            if i < mitad :
                xcorte[idx]=x1[i+1]
                ycorte[idx]=y1[i+1]
```

```
257                 else :
258                     if i < nofelements:
259                         xcorte[idx]=x1[i]
260                         ycorte[idx]=y1[i]
261                 idx=idx+1
262 #_____________________________________# p01
263         Mdb()
264         s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=2.0)
265         g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
266         s.setPrimaryObject(option=STANDALONE)
267         s.rectangle(point1=(-0.5, 0.0), point2=(0.5, 1.0))
268         p = mdb.models['Model-1'].Part(name='Part-1', dimensionality=TWO_D_PLANAR,
269             type=DEFORMABLE_BODY)
270         p = mdb.models['Model-1'].parts['Part-1']
271         p.BaseShell(sketch=s)
272         s.unsetPrimaryObject()
273         p = mdb.models['Model-1'].parts['Part-1']
274         del mdb.models['Model-1'].sketches['__profile__']
275 #_____________________________________# p02
276         p = mdb.models['Model-1'].parts['Part-1']
277         f1, e1, d2 = p.faces, p.edges, p.datums
278         t = p.MakeSketchTransform(sketchPlane=f1[0], sketchPlaneSide=SIDE1, origin=(
279             0.0, 0.0, 0.0))
280         s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=2.82,
281             gridSpacing=0.07, transform=t)
282         g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
283         s.setPrimaryObject(option=SUPERIMPOSE)
284         p = mdb.models['Model-1'].parts['Part-1']
285         p.projectReferencesOntoSketch(sketch=s, filter=COPLANAR_EDGES)
286         s.Line(point1=(0.0, -0.01), point2=(0.0+xpaux[0], ypaux[0]))
287         s.Line(point1=(0.0+xpaux[0], ypaux[0]), point2=(0.0+xpaux[1], ypaux[1]))
288         s.Line(point1=(0.0+xpaux[1], ypaux[1]), point2=(0.0+xpaux[2], ypaux[2]))
289         s.Line(point1=(0.0+xpaux[2], ypaux[2]), point2=(0.0+xpaux[3], ypaux[3]))
290         s.Line(point1=(0.0+xpaux[3], ypaux[3]), point2=(0.0+xpaux[4], ypaux[4]))
291         s.Line(point1=(0.0+xpaux[4], ypaux[4]), point2=(0.0, -0.01))
292         s.Line(point1=(0.5, 1), point2=(0.0+xpaux[1], ypaux[1]))
293         s.Line(point1=(-0.5, 1), point2=(0.0+xpaux[3], ypaux[3]))
294         s.Line(point1=(0.0, 0), point2=(0.0, 1))
295         s.Line(point1=(x2[0], y2[0]), point2=(x2[1], y2[1]))
296         s.Line(point1=(x2[1], y2[1]), point2=(x2[3], y2[3]))
297         s.Line(point1=(x2[3], y2[3]), point2=(x2[5], y2[5]))
298         s.Line(point1=(x2[5], y2[5]), point2=(x2[7], y2[7]))
299         s.Line(point1=(x2[7], y2[7]), point2=(x2[8], y2[8]))
300         s.CircleByCenterPerimeter(center=(0,0), point1=(rci, 0))
301         s.Line(point1=(x2[nofelements+1], y2[nofelements+1]), point2=(x2[nofelements-1], y2[nofelements-1]))
302         s.Line(point1=(x1[0] ,y1[0] ), point2=( x2[1], y2[1]))
303         s.Line(point1=(x2[1] ,y2[1] ), point2=( x1[1], y1[1]))
304         s.Line(point1=(x1[1] ,y1[1] ), point2=( x2[3], y2[3]))
305         s.Line(point1=(x2[3] ,y2[3] ), point2=( x1[2], y1[2]))
306         s.Line(point1=(x1[2] ,y1[2] ), point2=( x2[5], y2[5]))
307         s.Line(point1=(x2[5] ,y2[5] ), point2=( x1[3], y1[3]))
308         s.Line(point1=(x1[3] ,y1[3] ), point2=( x2[7], y2[7]))
309         s.Line(point1=(x2[7] ,y2[7] ), point2=( x1[4], y1[4]))
310         s.CircleByCenterPerimeter(center=(0,0), point1=(rco2, 0))
311         s.Line(point1=(x3[0] ,y3[0] ), point2=( x2[1], y2[1]))
312         s.Line(point1=(x2[1] ,y2[1] ), point2=( x3[1], y3[1]))
313         s.Line(point1=(x3[1] ,y3[1] ), point2=( x2[3], y2[3]))
314         s.Line(point1=(x2[3] ,y2[3] ), point2=( x3[3], y3[3]))
315         s.Line(point1=(x3[3] ,y3[3] ), point2=( x2[4], y2[3]))
316         s.Line(point1=(x2[4] ,y2[3] ), point2=( x3[5], y3[5]))
317         s.Line(point1=(x3[5] ,y3[5] ), point2=( x2[5], y2[5]))
318         s.Line(point1=(x2[5] ,y2[5] ), point2=( x3[7], y3[7]))
319         s.Line(point1=(x3[7] ,y3[7] ), point2=( x2[7], y2[7]))
320         s.Line(point1=(x2[7] ,y2[7] ), point2=( x3[8], y3[8]))
321         p = mdb.models['Model-1'].parts['Part-1']
322         f = p.faces
323         pickedFaces = f.getSequenceFromMask(mask=('[#1 ]', ), )
324         e, d1 = p.edges, p.datums
325         p.PartitionFaceBySketch(faces=pickedFaces, sketch=s)
326         s.unsetPrimaryObject()
327         del mdb.models['Model-1'].sketches['__profile__']
```

```
328
329             session.viewports['Viewport: 1'].view.setValues(nearPlane=2.75735,
330                 farPlane=2.8995, width=0.435438, height=0.194272,
331                 viewOffsetX=0.0172598, viewOffsetY=-0.4128)
332             session.viewports['Viewport: 1'].partDisplay.meshOptions.setValues(
333                 meshTechnique=OFF)
334             session.viewports['Viewport: 1'].partDisplay.geometryOptions.setValues(
335                 referenceRepresentation=ON)
336             p = mdb.models['Model-1'].parts['Part-1']
337             f = p.faces
338             p.RemoveFaces(faceList = f[2:3]+f[6:7], deleteCells=False)
339             session.viewports['Viewport: 1'].partDisplay.meshOptions.setValues(
340                 meshTechnique=ON)
341             session.viewports['Viewport: 1'].partDisplay.geometryOptions.setValues(
342                 referenceRepresentation=OFF)
343             p = mdb.models['Model-1'].parts['Part-1']
344             e = p.edges
345             pickedEdges = e.getSequenceFromMask(mask=('[#c3fffff  #7807e3 ]', ), )
346             p.seedEdgeByNumber(edges=pickedEdges, number=1, constraint=FIXED)
347             p = mdb.models['Model-1'].parts['Part-1']
348             e = p.edges
349             pickedEdges = e.getSequenceFromMask(mask=('[#20000000 #10000 ]', ), )
350             p.seedEdgeByNumber(edges=pickedEdges, number=3)
351             p = mdb.models['Model-1'].parts['Part-1']
352             e = p.edges
353             pickedEdges = e.getSequenceFromMask(mask=('[#0 #2010 ]', ), )
354             p.seedEdgeByNumber(edges=pickedEdges, number=int(math.ceil((1-alpha3)*3/2)))
355             p = mdb.models['Model-1'].parts['Part-1']
356             e = p.edges
357             pickedEdges = e.getSequenceFromMask(mask=('[#0 #801800 ]', ), )
358             p.seedEdgeByNumber(edges=pickedEdges, number=int(math.ceil((1-alpha3)*3)))
359             p = mdb.models['Model-1'].parts['Part-1']
360             p.seedPart(size=0.05, deviationFactor=0.1, minSizeFactor=0.1)
361             p = mdb.models['Model-1'].parts['Part-1']
362             f = p.faces
363             pickedRegions = f.getSequenceFromMask(mask=('[# fffffff  ]', ), )
364             p.setMeshControls(regions=pickedRegions, elemShape=TRI)
365             p = mdb.models['Model-1'].parts['Part-1']
366             p.generateMesh()
367
368             p = mdb.models['Model-1'].parts['Part-1']
369             f = p.faces
370             pickedRegions = f.getSequenceFromMask(mask=('[#4200000 ]', ), )
371             p.deleteMesh(regions=pickedRegions)
372             p = mdb.models['Model-1'].parts['Part-1']
373             f = p.faces
374             pickedRegions = f.getSequenceFromMask(mask=('[#4200000 ]', ), )
375             p.setMeshControls(regions=pickedRegions, technique=STRUCTURED)
376             p = mdb.models['Model-1'].parts['Part-1']
377             p.generateMesh()
378
379 #_____________________________________# p09
380             p = mdb.models['Model-1'].parts['Part-1']
381             p.generateMesh()
382             a = mdb.models['Model-1'].rootAssembly
383             session.viewports['Viewport: 1'].setValues(displayedObject=a)
384             session.viewports['Viewport: 1'].assemblyDisplay.setValues(
385                 optimizationTasks=OFF, geometricRestrictions=OFF, stopConditions=OFF)
386             a = mdb.models['Model-1'].rootAssembly
387             a.DatumCsysByDefault(CARTESIAN)
388             p = mdb.models['Model-1'].parts['Part-1']
389             a.Instance(name='Part-1-1', part=p, dependent=ON)
390             alpha1_s=str(alpha1).replace('.', '')
391             alpha2_s=str(alpha2).replace('.', '')
392             alpha3_s=str(alpha3).replace('.', '')
393             tipo1_s=str(tipo1)
394             tipo2_s=str(tipo2)
395             nofelements_s=str(nofelements)
396             nombre=nofelements_s+"E_tipo_"+tipo1_s+"_tipo_"+tipo2_s+"_"+alpha1_s[0:5]+"_"+alpha2_s[0:4]+"_"+alpha3_s
        [0:4]
397             mdb.Job(name=nombre, model='Model-1', description='', type=ANALYSIS,
```

```
                atTime=None, waitMinutes=0, waitHours=0, queue=None, memory=90,
                memoryUnits=PERCENTAGE, getMemoryFromAnalysis=True,
                 explicitPrecision =DOUBLE_PLUS_PACK, nodalOutputPrecision=FULL,
                echoPrint=OFF, modelPrint=OFF, contactPrint=OFF, historyPrint =OFF,
                userSubroutine='', scratch='', resultsFormat=ODB)
        mdb.jobs[nombre].writeInput(consistencyChecking=OFF)
        mdb.saveAs(
        pathName='C:/Users/Usuario/Desktop/TFG (Avances y Bibliografia )/Macros 2/'+nombre)

def Macro_4E_tipo_1_tipo_5b_alt_str():
    tipo1=1
    tipo2=5
#_____________________________________________________________ Cálculo de vértices de agujero y corona
    nofelements=4 # Introducir número de elementos
#
#    for alpha1 in wp.arange (0.05,0.75,0.05) :
#        alpha2min=alpha1*sqrt(2)
#        for alpha2 in wp.arange(alpha2min ,0.75,0.05) : # el segundo es 0.75
     sidelength=0.1
    xpaux=sidelength*wp.array ([1,0.5,0,-0.5,-1])
    ypaux=sidelength*wp.array([0,0.866025403784439,0.866025403784439,0.866025403784439,0])
    for alpha2 in wp.arange (0.1,0.5001,0.01) :
        m=1.75
        alpha3=m*alpha2
        p00=0.0013366475629475313
        p10=0.61196340890791623
        p01=-0.0023814962177391348
        p20=-0.01741752944192038
        p11=-0.011292193014912784
        p02=0.0053390577732086043
        alpha1=p00 + p10*alpha2 + p01*alpha3 + p20*alpha2**2 + p11*alpha2*alpha3 + p02*alpha3**2
        print[alpha1]
        print[alpha2]
        print[alpha3]
        N   = 2*nofelements
        n   = list(range(0, nofelements+1))
        n2= list(range(0, 2*nofelements+1))
        n3=list(range(0, 2*nofelements+1))
        rci= sidelength*alpha1
        rco= sidelength*alpha2
        rco2=sidelength*alpha3
        x1= list(range(0, nofelements+1))
        y1= list(range(0, nofelements+1))
        x2= list(range(0, 2*nofelements+1))
        y2= list(range(0, 2*nofelements+1))
        x3= list(range(0, 2*nofelements+1))
        y3= list(range(0, 2*nofelements+1))
        for i in range(0,nofelements+1):
            x1[i]=rci*cos(2*pi*n[i]/N)
            y1[i]=rci*sin(2*pi*n[i]/N)
        for i in range(0,2*nofelements+1):
            x2[i]=rco*cos(2*pi*n2[i]/(2*N))
            y2[i]=rco*sin(2*pi*n2[i]/(2*N))
        for i in range(0,2*nofelements+1):
            x3[i]=rco2*cos(2*pi*n3[i]/(2*N))
            y3[i]=rco2*sin(2*pi*n3[i]/(2*N))
#_____________________________________________________________ Definición xcorte e ycorte
        xcorte   = list(range(0, (nofelements*2)+1))
        ycorte   = list(range(0, (nofelements*2)+1))
        idx = 0
        mitad=nofelements/2
        for i in range(0,nofelements+1):
            xcorte[idx]=x2[i]
            ycorte[idx]=y2[i]
            idx=idx+1
            if i < mitad :
                xcorte[idx]=x1[i+1]
                ycorte[idx]=y1[i+1]
            else :
                if i < nofelements:
                    xcorte[idx]=x1[i]
```

```
                    ycorte[idx]=y1[i]
                idx=idx+1
#_____________________________________# p01
        Mdb()
        s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=2.0)
        g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
        s.setPrimaryObject(option=STANDALONE)
        s.rectangle(point1=(-0.5, 0.0), point2=(0.5, 1.0))
        p = mdb.models['Model-1'].Part(name='Part-1', dimensionality=TWO_D_PLANAR,
            type=DEFORMABLE_BODY)
        p = mdb.models['Model-1'].parts['Part-1']
        p.BaseShell(sketch=s)
        s.unsetPrimaryObject()
        p = mdb.models['Model-1'].parts['Part-1']
        del mdb.models['Model-1'].sketches['__profile__']
#_____________________________________# p02
        p = mdb.models['Model-1'].parts['Part-1']
        f1, e1, d2 = p.faces, p.edges, p.datums
        t = p.MakeSketchTransform(sketchPlane=f1[0], sketchPlaneSide=SIDE1, origin=(
            0.0, 0.0, 0.0))
        s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=2.82,
            gridSpacing=0.07, transform=t)
        g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
        s.setPrimaryObject(option=SUPERIMPOSE)
        p = mdb.models['Model-1'].parts['Part-1']
        p.projectReferencesOntoSketch(sketch=s, filter=COPLANAR_EDGES)
        s.Line(point1=(0.0, -0.01), point2=(0.0+xpaux[0], ypaux[0]))
        s.Line(point1=(0.0+xpaux[0], ypaux[0]), point2=(0.0+xpaux[1], ypaux[1]))
        s.Line(point1=(0.0+xpaux[1], ypaux[1]), point2=(0.0+xpaux[2], ypaux[2]))
        s.Line(point1=(0.0+xpaux[2], ypaux[2]), point2=(0.0+xpaux[3], ypaux[3]))
        s.Line(point1=(0.0+xpaux[3], ypaux[3]), point2=(0.0+xpaux[4], ypaux[4]))
        s.Line(point1=(0.0+xpaux[4], ypaux[4]), point2=(0.0, -0.01))
        s.Line(point1=(0.5, 1), point2=(0.0+xpaux[1], ypaux[1]))
        s.Line(point1=(-0.5, 1), point2=(0.0+xpaux[3], ypaux[3]))
        s.Line(point1=(0.0, 0), point2=(0.0, 1))
        s.CircleByCenterPerimeter(center=(0,0), point1=(rci, 0))
        s.ArcByCenterEnds(center=(0,0), point1=(-rco, 0),
            point2=(x2[nofelements+1], y2[nofelements+1]), direction=CLOCKWISE)
        s.ArcByCenterEnds(center=(0,0), point1=(rco, 0),
            point2=(x2[nofelements-1], y2[nofelements-1]), direction=COUNTERCLOCKWISE)
        s.Line(point1=(x2[nofelements+1], y2[nofelements+1]), point2=(x2[nofelements-1], y2[nofelements-1]))
        s.Line(point1=(x1[0] ,y1[0] ), point2=( x2[1], y2[1]))
        s.Line(point1=(x2[1] ,y2[1] ), point2=( x1[1], y1[1]))
        s.Line(point1=(x1[1] ,y1[1] ), point2=( x2[3], y2[3]))
        s.Line(point1=(x2[3] ,y2[3] ), point2=( x1[2], y1[2]))
        s.Line(point1=(x1[2] ,y1[2] ), point2=( x2[5], y2[5]))
        s.Line(point1=(x2[5] ,y2[5] ), point2=( x1[3], y1[3]))
        s.Line(point1=(x1[3] ,y1[3] ), point2=( x2[7], y2[7]))
        s.Line(point1=(x2[7] ,y2[7] ), point2=( x1[4], y1[4]))
        s.CircleByCenterPerimeter(center=(0,0), point1=(rco2, 0))
        s.Line(point1=(x3[0] ,y3[0] ), point2=( x2[1], y2[1]))
        s.Line(point1=(x2[1] ,y2[1] ), point2=( x3[1], y3[1]))
        s.Line(point1=(x3[1] ,y3[1] ), point2=( x2[3], y2[3]))
        s.Line(point1=(x2[3] ,y2[3] ), point2=( x3[3], y3[3]))
        s.Line(point1=(x3[3] ,y3[3] ), point2=( x2[4], y2[3]))
        s.Line(point1=(x2[4] ,y2[3] ), point2=( x3[5], y3[5]))
        s.Line(point1=(x3[5] ,y3[5] ), point2=( x2[5], y2[5]))
        s.Line(point1=(x2[5] ,y2[5] ), point2=( x3[7], y3[7]))
        s.Line(point1=(x3[7] ,y3[7] ), point2=( x2[7], y2[7]))
        s.Line(point1=(x2[7] ,y2[7] ), point2=( x3[8], y3[8]))
        p = mdb.models['Model-1'].parts['Part-1']
        f = p.faces
        pickedFaces = f.getSequenceFromMask(mask=('[#1 ]', ), )
        e, d1 = p.edges, p.datums
        p.PartitionFaceBySketch(faces=pickedFaces, sketch=s)
        s.unsetPrimaryObject()
        del mdb.models['Model-1'].sketches['__profile__']

        p = mdb.models['Model-1'].parts['Part-1']
        f1 = p.faces
        p.RemoveFaces(faceList = f1[2:3]+f1[6:7], deleteCells=False)
```

```
            session.viewports['Viewport: 1'].partDisplay.meshOptions.setValues(
                meshTechnique=ON)
            session.viewports['Viewport: 1'].partDisplay.geometryOptions.setValues(
                referenceRepresentation=OFF)
            session.viewports['Viewport: 1'].view.setValues(nearPlane=2.78418,
                farPlane=2.87267, width=0.348169, height=0.151909,
                viewOffsetX=-0.00153783, viewOffsetY=-0.434711)
            p = mdb.models['Model-1'].parts['Part-1']
            e = p.edges
            pickedEdges = e.getSequenceFromMask(mask=('[#c3ffffff #7807e3 ]', ), )
            p.seedEdgeByNumber(edges=pickedEdges, number=1, constraint=FIXED)
            p = mdb.models['Model-1'].parts['Part-1']
            e = p.edges
            pickedEdges = e.getSequenceFromMask(mask=('[#20000000 #10000 ]', ), )
            p.seedEdgeByNumber(edges=pickedEdges, number=3)
            p = mdb.models['Model-1'].parts['Part-1']
            e = p.edges
            pickedEdges = e.getSequenceFromMask(mask=('[#0 #803810 ]', ), )
            p.seedEdgeByNumber(edges=pickedEdges, number=1)
            p = mdb.models['Model-1'].parts['Part-1']
            p.seedPart(size=0.05, deviationFactor=0.1, minSizeFactor=0.1)
            p = mdb.models['Model-1'].parts['Part-1']
            f = p.faces
            pickedRegions = f.getSequenceFromMask(mask=('[#ffffff ]', ), )
            p.setMeshControls(regions=pickedRegions, elemShape=TRI)
            p = mdb.models['Model-1'].parts['Part-1']
            p.generateMesh()

            p = mdb.models['Model-1'].parts['Part-1']
            f = p.faces
            pickedRegions = f.getSequenceFromMask(mask=('[#4200000 ]', ), )
            p.deleteMesh(regions=pickedRegions)
            p = mdb.models['Model-1'].parts['Part-1']
            f = p.faces
            pickedRegions = f.getSequenceFromMask(mask=('[#4200000 ]', ), )
            p.setMeshControls(regions=pickedRegions, technique=STRUCTURED)
            p = mdb.models['Model-1'].parts['Part-1']
            p.generateMesh()


#_____________________________________# p09
            p = mdb.models['Model-1'].parts['Part-1']
            p.generateMesh()
            a = mdb.models['Model-1'].rootAssembly
            session.viewports['Viewport: 1'].setValues(displayedObject=a)
            session.viewports['Viewport: 1'].assemblyDisplay.setValues(
                optimizationTasks=OFF, geometricRestrictions=OFF, stopConditions=OFF)
            a = mdb.models['Model-1'].rootAssembly
            a.DatumCsysByDefault(CARTESIAN)
            p = mdb.models['Model-1'].parts['Part-1']
            a.Instance(name='Part-1-1', part=p, dependent=ON)
            alpha1_s=str(alpha1).replace('.', '')
            alpha2_s=str(alpha2).replace('.', '')
            alpha3_s=str(alpha3).replace('.', '')
            tipo1_s=str(tipo1)
            tipo2_s=str(tipo2)
            nofelements_s=str(nofelements)
            nombre=nofelements_s+"E_tipo_"+tipo1_s+"_tipo_"+tipo2_s+"_"+alpha1_s[0:5]+"_"+alpha2_s[0:4]+"_"+alpha3_s
        [0:4]
            mdb.Job(name=nombre, model='Model-1', description='', type=ANALYSIS,
                atTime=None, waitMinutes=0, waitHours=0, queue=None, memory=90,
                memoryUnits=PERCENTAGE, getMemoryFromAnalysis=True,
                explicitPrecision=DOUBLE_PLUS_PACK, nodalOutputPrecision=FULL,
                echoPrint=OFF, modelPrint=OFF, contactPrint=OFF, historyPrint=OFF,
                userSubroutine='', scratch='', resultsFormat=ODB)
            mdb.jobs[nombre].writeInput(consistencyChecking=OFF)
            mdb.saveAs(
            pathName='C:/Users/Usuario/Desktop/TFG (Avances y Bibliografia )/Macros 2/'+nombre)
```

Script 9.12 Study 2 Abaqus Macro.

Study 3

```
def Macro_6E_tipo_2_tipo_5a():
    tipo1=2
    tipo2=5
#_____________________________________________________________ Cálculo de vértices de agujero y corona
    nofelements=6 # Introducir número de elementos
#
#     for alpha1 in wp.arange (0.05,0.75,0.05) :
#         alpha2min=alpha1*sqrt(2)
#         for alpha2 in wp.arange(alpha2min ,0.75,0.05) : # el segundo es 0.75
     sidelength =0.1
    xpaux=sidelength*wp.array ([1,0.5,0,-0.5,-1])
    ypaux=sidelength*wp.array([0,0.866025403784439,0.866025403784439,0.866025403784439,0])
#     for alpha1 in wp.arange (0.05,0.75,0.05) :
#         alpha2min=alpha1*sqrt(2)
#         for alpha2 in wp.arange(alpha2min ,0.75,0.05) : # el segundo es 0.75
    for sidelength in wp.arange (0.05,0.501,0.01) :
        xpaux=sidelength*wp.array ([1,0.5,0,-0.5,-1])
        ypaux=sidelength*wp.array([0,0.866025403784439,0.866025403784439,0.866025403784439,0])
        alpha1=0.342192051008172
        m=2.25
        alpha3=m*alpha1
        p00=0.092033308506444755
        p10=1.4444178518405617
        p01=-0.2205000738061571
        p20=0.055413078587417446
        p11=0.30285177151819498
        p02=0.12068664377057438
        alpha2=p00 + p10*alpha1 + p01*alpha3 + p20*alpha1**2 + p11*alpha1*alpha3 + p02*alpha3**2
        N   = 2*nofelements
        n   = list (range(0, nofelements+1))
        n2= list (range(0, nofelements+1))
        n3= list (range(0, 2*nofelements+1))
        rci = sidelength*alpha1
        rco= sidelength*alpha2
        rco2= sidelength*alpha3
        x1= list (range(0, nofelements+1))
        y1= list (range(0, nofelements+1))
        x2= list (range(0, nofelements+1))
        y2= list (range(0, nofelements+1))
        x3= list (range(0, 2*nofelements+1))
        y3= list (range(0, 2*nofelements+1))
        for i in range(0,nofelements+1):
            x1[i]= rci*cos(2*pi*n[i]/N)
            y1[i]= rci*sin(2*pi*n[i]/N)
        for i in range(0,nofelements+1):
            x2[i]=rco*cos(2*pi*n2[i]/(N))
            y2[i]=rco*sin(2*pi*n2[i]/(N))
        for i in range(0,2*nofelements+1):
            x3[i]=rco2*cos(2*pi*n3[i]/(2*N))
            y3[i]=rco2*sin(2*pi*n3[i]/(2*N))
#_____________________________________________________________ Definición xcorte e ycorte

        xcorte   = list (range(0, (nofelements*2)+1))
        ycorte   = list (range(0, (nofelements*2)+1))
        idx = 0
        mitad=nofelements/2
        for i in range(0,nofelements+1):
            xcorte[idx]=x2[i]
            ycorte[idx]=y2[i]
            idx=idx+1
            if i < mitad :
                xcorte[idx]=x1[i+1]
                ycorte[idx]=y1[i+1]
            else :
                if i < nofelements:
                    xcorte[idx]=x1[i]
                    ycorte[idx]=y1[i]
            idx=idx+1
```

```
#_____________________________________# p01
        Mdb()
        session.viewports['Viewport: 1'].setValues(displayedObject=None)
        s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=2.0)
        g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
        s.setPrimaryObject(option=STANDALONE)
        s.rectangle(point1=(-0.5, 0.0), point2=(0.5, 1.0))
        p = mdb.models['Model-1'].Part(name='Part-1', dimensionality=TWO_D_PLANAR,
            type=DEFORMABLE_BODY)
        p = mdb.models['Model-1'].parts['Part-1']
        p.BaseShell(sketch=s)
        s.unsetPrimaryObject()
        p = mdb.models['Model-1'].parts['Part-1']
        session.viewports['Viewport: 1'].setValues(displayedObject=p)
        del mdb.models['Model-1'].sketches['__profile__']
#_____________________________________# p02
        p = mdb.models['Model-1'].parts['Part-1']
        f1, e1, d2 = p.faces, p.edges, p.datums
        t = p.MakeSketchTransform(sketchPlane=f1[0], sketchPlaneSide=SIDE1, origin=(
            0.0, 0.0, 0.0))
        s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=2.82,
            gridSpacing=0.07, transform=t)
        g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
        s.setPrimaryObject(option=SUPERIMPOSE)
        p = mdb.models['Model-1'].parts['Part-1']
        p.projectReferencesOntoSketch(sketch=s, filter=COPLANAR_EDGES)
        s.Line(point1=(0.0, -0.01), point2=(0.0+xpaux[0], ypaux[0]))
        s.Line(point1=(0.0+xpaux[0], ypaux[0]), point2=(0.0+xpaux[1], ypaux[1]))
        s.Line(point1=(0.0+xpaux[1], ypaux[1]), point2=(0.0+xpaux[2], ypaux[2]))
        s.Line(point1=(0.0+xpaux[2], ypaux[2]), point2=(0.0+xpaux[3], ypaux[3]))
        s.Line(point1=(0.0+xpaux[3], ypaux[3]), point2=(0.0+xpaux[4], ypaux[4]))
        s.Line(point1=(0.0+xpaux[4], ypaux[4]), point2=(0.0, -0.01))
        s.Line(point1=(0.5, 1), point2=(0.0+xpaux[1], ypaux[1]))
        s.Line(point1=(-0.5, 1), point2=(0.0+xpaux[3], ypaux[3]))
        s.Line(point1=(0.0, 0), point2=(0.0, 1))
        s.CircleByCenterPerimeter(center=(0,0), point1=(rci, 0))
        s.CircleByCenterPerimeter(center=(0,0), point1=(rco, 0))
        s.Line(point1=(x1[0] ,y1[0] ), point2=( x2[1], y2[1]))
        s.Line(point1=(x1[1] ,y1[1] ), point2=( x2[1], y2[1]))
        s.Line(point1=(x1[1] ,y1[1] ), point2=( x2[2], y2[2]))
        s.Line(point1=(x1[2] ,y1[2] ), point2=( x2[2], y2[2]))
        s.Line(point1=(x1[2] ,y1[2] ), point2=( x2[3], y2[3]))
        s.Line(point1=(x1[3] ,y1[3] ), point2=( x2[3], y2[3]))
        s.Line(point1=(x2[3] ,y2[3] ), point2=( x1[4], y1[4]))
        s.Line(point1=(x2[4] ,y2[4] ), point2=( x1[4], y1[4]))
        s.Line(point1=(x2[4] ,y2[4] ), point2=( x1[5], y1[5]))
        s.Line(point1=(x2[5] ,y2[5] ), point2=( x1[5], y1[5]))
        s.Line(point1=(x2[5] ,y2[5] ), point2=( x1[6], y1[6]))
        s.ArcByCenterEnds(center=(0,0), point1=(rco2, 0),
            point2=(x3[nofelements+1], y3[nofelements+1]), direction=CLOCKWISE)
        s.ArcByCenterEnds(center=(0,0), point1=(rco2, 0),
            point2=(x3[nofelements-1], y3[nofelements-1]), direction=COUNTERCLOCKWISE)
        s.Line(point1=(x3[nofelements+1], y3[nofelements+1]), point2=(x3[nofelements-1], y3[nofelements-1]))
        s.Line(point1=(x2[0] ,y2[0] ), point2=( x3[1] , y3[1] ))
        s.Line(point1=(x3[1] ,y3[1] ), point2=( x2[1] , y2[1] ))
        s.Line(point1=(x2[1] ,y2[1] ), point2=( x3[3] , y3[3] ))
        s.Line(point1=(x3[3] ,y3[3] ), point2=( x2[2] , y2[2] ))
        s.Line(point1=(x2[2] ,y2[2] ), point2=( x3[5] , y3[5] ))
        s.Line(point1=(x3[5] ,y3[5] ), point2=( x2[3] , y2[3] ))
        s.Line(point1=(x2[3] ,y2[3] ), point2=( x3[7] , y3[7] ))
        s.Line(point1=(x3[7] ,y3[7] ), point2=( x2[4] , y2[4] ))
        s.Line(point1=(x2[4] ,y2[4] ), point2=( x3[9] , y3[9] ))
        s.Line(point1=(x3[9] ,y3[9] ), point2=( x2[5] , y2[5] ))
        s.Line(point1=(x2[5] ,y2[5] ), point2=( x3[11], y3[11]))
        s.Line(point1=(x3[11] ,y3[11] ), point2=( x2[6], y2[6]))
        p = mdb.models['Model-1'].parts['Part-1']
        f = p.faces
        pickedFaces = f.getSequenceFromMask(mask=('[#1 ]', ), )
        e, d1 = p.edges, p.datums
        p.PartitionFaceBySketch(faces=pickedFaces, sketch=s)
        s.unsetPrimaryObject()
```

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

#_____________________________________# p09

            session.viewports['Viewport: 1'].view.setValues(nearPlane=2.70964,
                farPlane=2.94722, width=0.729638, height=0.318346,
                viewOffsetX=0.00364796, viewOffsetY=-0.366037)
            session.viewports['Viewport: 1'].partDisplay.meshOptions.setValues(
                meshTechnique=OFF)
            session.viewports['Viewport: 1'].partDisplay.geometryOptions.setValues(
                referenceRepresentation=ON)
            p = mdb.models['Model-1'].parts['Part-1']
            session.viewports['Viewport: 1'].setValues(displayedObject=p)
            p = mdb.models['Model-1'].parts['Part-1']
            f = p.faces
            p.RemoveFaces(faceList = f[29:30]+f[32:33], deleteCells=False)
            session.viewports['Viewport: 1'].view.setValues(nearPlane=2.75579,
                farPlane=2.90106, width=0.446368, height=0.194163,
                viewOffsetX=0.0143654, viewOffsetY=-0.4)
            session.viewports['Viewport: 1'].partDisplay.meshOptions.setValues(
                meshTechnique=ON)
            session.viewports['Viewport: 1'].partDisplay.geometryOptions.setValues(
                referenceRepresentation=OFF)
            p = mdb.models['Model-1'].parts['Part-1']
            e = p.edges
            pickedEdges = e.getSequenceFromMask(mask=('[#f807ff80 # ffffff7 ]', ), )
            p.seedEdgeByNumber(edges=pickedEdges, number=1, constraint=FIXED)
            p = mdb.models['Model-1'].parts['Part-1']
            p.seedPart(size=0.05, deviationFactor=0.1, minSizeFactor=0.1)
            p = mdb.models['Model-1'].parts['Part-1']
            f = p.faces
            pickedRegions = f.getSequenceFromMask(mask=('[# fffffff ]', ), )
            p.setMeshControls(regions=pickedRegions, elemShape=TRI)
            p = mdb.models['Model-1'].parts['Part-1']
            p.generateMesh()

#_____________________________________# p09

            a = mdb.models['Model-1'].rootAssembly
            session.viewports['Viewport: 1'].setValues(displayedObject=a)
            session.viewports['Viewport: 1'].assemblyDisplay.setValues(
                optimizationTasks=OFF, geometricRestrictions=OFF, stopConditions=OFF)
            a = mdb.models['Model-1'].rootAssembly
            a.DatumCsysByDefault(CARTESIAN)
            p = mdb.models['Model-1'].parts['Part-1']
            a.Instance(name='Part-1-1', part=p, dependent=ON)
            alpha1_s=str(alpha1*sidelength).replace('.', '')
            alpha2_s=str(alpha2).replace('.', '')
            alpha3_s=str(alpha3).replace('.', '')
            tipo1_s=str(tipo1)
            tipo2_s=str(tipo2)
            nofelements_s=str(nofelements)
            nombre=nofelements_s+"E_tipo_"+tipo1_s+"_tipo_"+tipo2_s+"_"+alpha1_s[0:5]+"_"+alpha2_s[0:4]+"_"+alpha3_s
        [0:4]
            mdb.Job(name=nombre, model='Model-1', description='', type=ANALYSIS,
                atTime=None, waitMinutes=0, waitHours=0, queue=None, memory=90,
                memoryUnits=PERCENTAGE, getMemoryFromAnalysis=True,
                explicitPrecision=DOUBLE_PLUS_PACK, nodalOutputPrecision=FULL,
                echoPrint=OFF, modelPrint=OFF, contactPrint=OFF, historyPrint=OFF,
                userSubroutine='', scratch='', resultsFormat=ODB)
            mdb.jobs[nombre].writeInput(consistencyChecking=OFF)
            mdb.saveAs(
                pathName='C:/Users/Usuario/Desktop/TFG (Avances y Bibliografia )/Macros 3/'+nombre)
```

Script 9.13 Study 3 Abaqus Macro.

Figures Index

Index of tables

Index of scripts

References

[1] A. Agrawal and A.M. Karlsson, *On the reference length and mode mixity for a bimaterial interface*, Journal of Engineering Materials and Technology **129** (2007), 580–587.

[2] J.E. Akin, *The generation of elements with singularities*, International journal for numerical methods in engineering **11** (1977), 85–98.

[3] J. Alberty, C. Carstensen, and A. Funken, *Remarks around 50 lines of Matlab: short finite element implementation*, Numerical Algorithms **20** (1999), 117–137.

[4] S. Jiménez Alfaro, *Singular elastic solutions for corners and cracks with spring boundary conditions*, Final Master Project, Universidad de Sevilla, 2020.

[5] P. Ariza Moreno and J.A. Sanz Herrera, *Finite elements method (academical practice), (in spanish)*, 2009/2010.

[6] Roshdy S. Barsoum, *On the use ofisoparametric finite elements in linear fracture mechanics* , INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING **10** (1976), 25–37.

[7] S.E. Benzley, *Representation of singularities with isoparametric finite elements*, International journal for numerical methods in engineering **8** (1974), 537–545.

[8] F. París Carballo, *Elasticity Theory (Second Edition) (in Spanish)*, Group of Elasticity and Strength of Materials, Universidad de Sevilla, 2000.

[9] L. De Lorenzis and C. Maurini, *MecFract Seminar 2021*, Basics of computational methods for fracture, 2021.

[10] I. Fuenzalida-Henriquez, E. Castillo-Ibarra, and Hinojosa J., *Global local analysis with Robin parameters: applications to crack propagation in 2D and 3D models*, 14th World Congress on Computational Mechanics (WCCM) ECCOMAS Congress 2020 **774** (2021), 355–360.

[11] Nitin S. Gokhale, Sanjay S. Deshpande, Sanjeev V. Bedekar, and N. Thite Anand, *Practical Finite Element Analysis*, ch. 7.9.

[12] R.D. Henshell and K.G. Shaw, *Crack tip finite elements are unnecesary*, International journal for numerical methods in engineering **9** (1975), 495–507.

[13] T.J.R. Hughes and J.E. Akin, *Techniques for developing 'special'finite element shape functiones with particular reference to singularities*, International journal for numerical methods in engineering **15** (1980), 733–751.

[14] Kamnle, M., https://www.quora.com/Why-is-the-global-stiffness-matrix-banded-in-FEM , 28th May 2018.

[15] M. F. Kanninen, *An augmented double cantilever beam model for studying crack propagation and arrest*, International Journal of Fracture **9** (1973), 83–92.

[16] Paul M. Kurowski, *Finite Element Analysis For Design Engineers*, ch. 5.3.1.

[17] M. Romero Laborda, *Implementation of special singular finite elements for cracks in adhesive interfaces (in Spanish)*, Final Degree Project, Universidad de Sevilla, 2020.

[18] S. Lenci, *Analysis of a crack at a weak interface*, International Journal of Fracture **108** (2001), 275–290.

[19] I.L. Lim, I.W. Johnston, and S.K. Choi, *Application of singular quadratic distorted isoparametric elements in linear fracture mechanics*, International journal for numerical methods in engineering **36** (1993), 2473–2499.

[20] G.R. Liu, *The Finite Element Method: Practical Course*, ch. 11.4.2.

[21] Muñoz-Reja M., Távara L., and V. Mantič, *Convergence of the BEM Solution Applied to the CCFFM for LEBIM*, Trans Tech Publications **774** (2018), 355–360.

[22] V. Mantič, Vázquez-Sánchez A., Romero-Laborda M., Muńoz-Reja M., Jiménez-Alfaro S., and Távara L., *New crack tip element for logarithmic stress-singularity of cracks on spring interfaces in mode III*, Submitted (Submitted).

[23] V. Mantič and F. París Carballo, *Plasticity at the neighbourhood of the crack tip*, 2020.

[24] M. Muñoz Reja, L. Távara, V. Mantič, and P. Cornetti, *Crack onset and propagation at fibre-matrix elastic interfaces under biaxial loading using finite fracture mechanics*, Composites Part A **82** (2016), 267–278.

[25] F. París Carballo, *Lectures on Solid Mechanics (in Spanish)*, Chapter 1: Principles and Theorems of Elasticity, 2020.

[26] F. París Carballo and García García I., *Lectures on Solid Mechanics (in Spanish)*, Chapter 4: Non-isotropic behavior, 2020.

[27] V. Mantic S. Jiménez-Alfaro, V. Villalba, *Singular elastic solutions in corners with spring boundary conditions under anti-plane shear*, International Journal of Fracture (2020), 197–220, *https://doi.org/10.1007/s10704-020-00443-5*.

[28] V.E. Saouma and D. Schwemmer, *Numerical evaluation of the quarter-point crack tip element*, Int. j. numer. methods eng. **12** (1984), 1629–1641.

[29] M. Smith, *ABAQUS/Standard User's Manual, Version 6.9*, Dassault Systèmes Simulia Corp, Providence, RI, 2009.

[30] B. Szabo and I. Babuska, *The finite element analysis*, ch. 7: Computation of stiffness matrices and load vector for two dimensional elastostatic problems, 1991.

[31] B. Szabo and I. Babuska, *The finite element analysis*, ch. 8: Potential flow problems, 1991.

[32] B. Szabo and I. Babuska, *The finite element analysis*, ch. 4.1/4.2/4.3, 2011.

[33] L. Távara, V. Mantič, E. Graciani, J. Cañas, and F. París, *Analysis of a crack in a thin adhesive layer between orthotropic materials applied to composite interlaminar fracture toughness test*, Composites 2009, 2nd ECCOMAS Thematic Conference on the Mechanical Response of Composites, London, UK (2009).

[34] O.C. Zienkiewicz, R.L. Taylor, and J.Z. Zhu, *The finite element method: Its basis and fundamentals*, fifth ed., Butterworth-Heinemann, 2005.