Master's Degree Final Project
Industrial Engineering

Implementation in MATLAB of the Iso-Geometric Boundary Elements Method for the resolution of 2D anisotropic elastostatic problems

Author:

Alberto Sánchez-Reyes González

Tutor:

Andrés Sáez Pérez

Continuum Mechanics and Theory of Structures
School of Engineering
Universidad de Sevilla

Seville, 2020

Master's Degree Final Project
Industrial Engineering

Implementation in MATLAB of the Iso-Geometric Boundary Elements Method for the resolution of 2D anisotropic elastostatic problems

Author:

Alberto Sánchez-Reyes González

Tutors:

Andrés Sáez Pérez

Full Professor

Continuum Mechanics and Theory of Structures
School of Engineering
Universidad de Sevilla

Seville, 2020

Master's Degree Final Project: Implementation in MATLAB of the Iso-Geometric Boundary Elements Method for the resolution of 2D anisotropic elastostatic problems

Author: Alberto Sánchez-Reyes González
Tutors: Andrés Sáez Pérez

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:

Abstract

In this work, the implementation of the Isogeometric Boundary Element Method (IGABEM) in MATLAB for the resolution of bi-dimensional anisotropic elastostatic problems is presented. This builds on the degree's final project of this author, which studied the IGABEM for the isotropic case

Firstly, some basics about anisotropic elasticity are given, mostly to clarify nomenclature. Secondly, an introduction to the traditional Boundary Element Method (BEM) is offered, explaining the physics behind it and later particularizing the equations for its numerical implementation. The most important details, or those that take relevance when compared to the IGABEM, are commented. Subsequently, the same process is repeated with the IGABEM, adapting all the equations to this method's features. The implemented MATLAB code is briefly commented, its difficulties and limitations highlighted. Finally, a comparison between the two methods is done. Analysis are carried out in two elastic problems, one orthotropic and one purely anisotropic, and the results are tested. This does not only show the difference in accuracy but also serves to spot better the differences between these two methods and the advantages of one over the other.

Finally, some options for further work are discussed. Together with the ideas, an outline of how they would be made and why they would be useful or interesting is made.

Table of Contents

List of Figures

List of Codes

1 Introduction

In this chapter, the motivation of this work will be explained, starting with a brief look into anisotropic materials, followed by an explanation of the usefulness of the Isogeometric Boundary Element Method (IGABEM). Then, an overall view of this document will be given, describing the steps taken on each chapter.

1.1 Motivation

Anisotropic materials are constantly gaining importance as new composites are developed and its use becomes more widely spread. Although there are plenty of anisotropic materials in nature, such as many woods and crystals, it is thanks to the recent year's development of man-made materials that anisotropy has become an area of special interest to researchers. Figure 1.1 shows the clearly increasing influence of composite materials in modern day aviation, which is but one of the many fields on which they are used. Furthermore, their influence is very likely to continue in their upward trend as research finds new opportunities and production methods improve.

It is also interesting to note that even materials we would consider isotropic, such as aluminum and its alloys, have some small degree of anisotropy that can be magnified through treatments as rolling, extrusion, etc [11]. This makes the study of anisotropic materials even more compelling, since there are cases on which isotropic elasticity will not suffice.

Among the aforementioned anisotropic crystals, piezoelectric materials hold special significance inside the group. Not only do they posses great application potential, but also they make the Boundary Element Method (BEM) extremely advantageous for their study as one of their characteristics is the propensity to develop cracks. This happens both for their intrinsic fragility as for the processes they are subjected to in order to make them piezoelectric [7]. The BEM is an algorithm for solving the elastic problem that has been proven to be an interesting alternative to the better known Finite Element Method (FEM) in several cases, specifically in crack analysis, advantage that is shared with the method under study in this work, the Iso-Geometric Boundary Element Method (IGABEM).

The IGABEM comes in to close the gap that exists in modern day engineering between Computer Aided Design (CAD) and analysis executions. As a basis, designers use CAD to create the desired item, with a wide range of possibilities going from simple forms to complex, highly detailed objects. Then, a structural analysis must be carried out by Computer Aided Engineering (CAE) either with FEM, BEM or any other. These two steps are not independent one of the other, with huge impact of the analysis results on the final shape and the requirement of several iterations to reach a conclusion. However, that transition is not that easy, since the CAD model must be transformed into a form suitable for the analysis by the discretization of the domain (or boundary) meshing it. This step is the one that consumes more time throughout the procedure, sometimes taking up to 80 % of the total time and constituting a bottle-neck to the process (see Figure 1.2).

The IGABEM aims to overcome this hurdle by skipping, or at least reducing the meshing. The idea is to employ the same CAD functions programs use to discretize the boundary (in this work the most common will be used, called NURBS from Non-Uniform Rational B-Splines) to also approximate the unknown fields. This replaces the polynomial shape functions handled in traditional BEM.

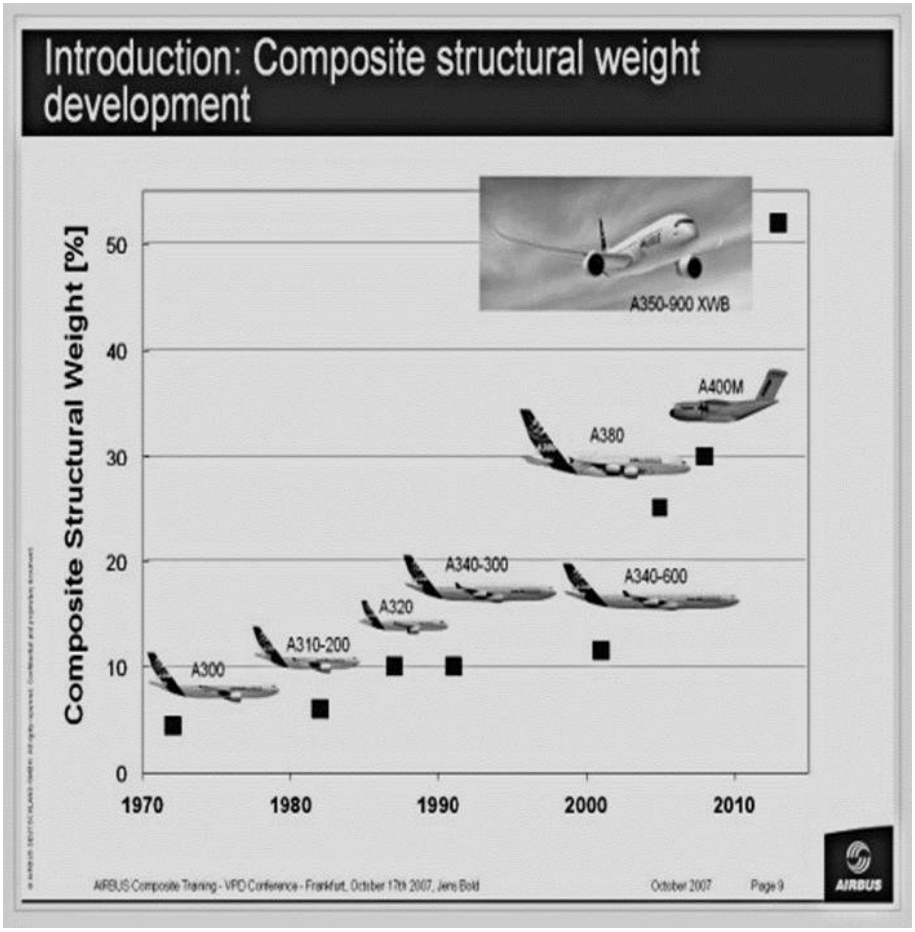


Figure 1.1 Increase in composite material use in aircraft over past years. Figure taken from [10].

At the cost of having to evaluate more complex functions, several advantages are gained. First and foremost, as has already been stated, the role of the mesh is done by the CAD program in the designing of the part. Besides, this way allows the analysis to be carried out while exactly preserving the geometry of the problem. This takes great importance in problems where the accuracy of the boundary is critically sensitive to geometrical imperfections, such as shell buckling or sliding contact between solids (see Figure 1.3). However, one of the biggest shortcomings of this method is related to the refinement of the analysis. As will be shown in 6.1, sometimes the geometry will be simple (and able to be represented with a small number of functions) while the unknowns fields will not. In such cases, a refinement must be implemented. Nonetheless, the way in which NURBS are built does not allow local refinement, enforcing a global, expensive refinement. An improvement in this field would be the implementation of T-Splines (a superset of NURBS) or PHT-Splines which allow local refinement, solving this deficiency and expanding the method's adaptivity. For further information, [2] covers T-Splines more deeply and [21] does the same for PHT-splines.

This work is centered on two MATLAB codes the author has developed for solving elastic bidimensional problems applying both methods (BEM and IGABEM) to anisotropic materials. The greater part of the document focused on explaining the theory that allow said methods to work; the rest covers the structure of the code and the numerical results it has yielded, which are compared.

1.2 Document organization

Chapter 1 is this very section, which makes a case for the value of this work and offers an explained index of its contents.

In Chapter 2 some of the ruling equations in anisotropic elasticity will be presented. It will not be an extensive explanation and it will be limited to the equations that are needed to be known in order to understand

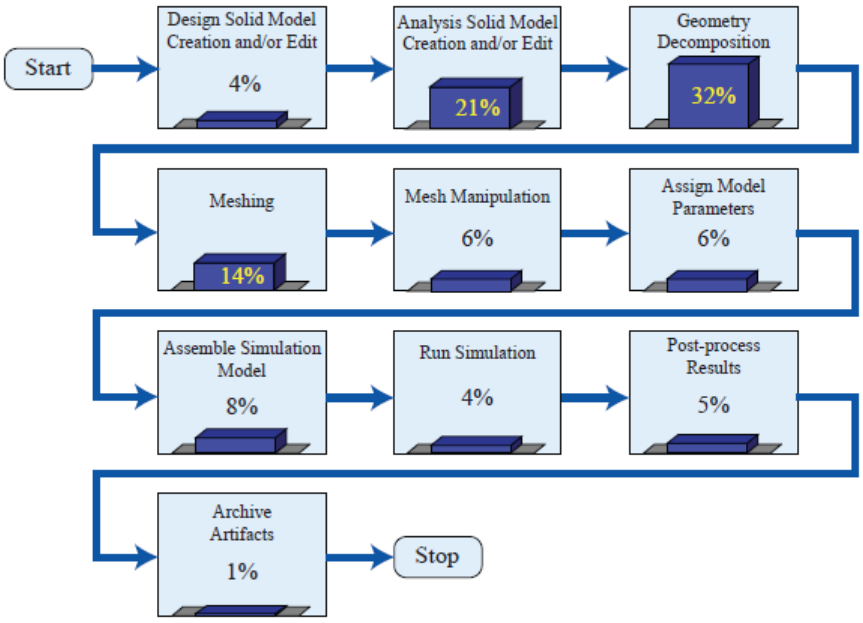


Figure 1.2 Estimation of the relative time costs of each component of the model generation and analysis process at Sandia National Laboratories. Figure taken from [3].

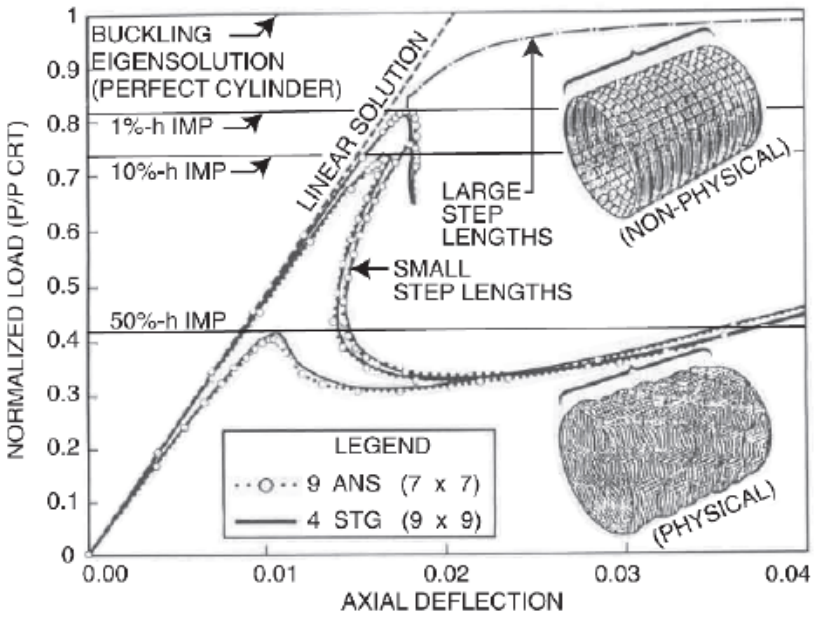


Figure 1.3 Buckling load of a cylindrical shell with geometrical imperfections. Figure taken from [3].

the following chapters.

In Chapter 3 the theoretical explanation behind the equations used in the BEM will be fully developed. The only requisite to the fully understanding of the chapter is to be familiar with the elasticity constitutive equations stated in the prior chapter.

In Chapter 4, the traditional BEM will be discussed. First an overall view of how the system of equations is obtained and solved will be presented, then some more specific details about the integrations will be given. For the sake of completeness, the calculation of the fundamental solutions and the free terms will

be included (despite the latter being optional for the execution of the method.) Additionally there will be some comments on how isotropic materials should be treated, being a particular case of the bigger group of anisotropic materials. It will finally introduce the MATLAB program the author has developed without deepening into details. A flux diagram will be attached to show the functioning of the code.

In Chapter 5 the IGABEM will be examined. This chapter will start with an introduction to how B-SPLINES and NURBS work. Then the modification of the elasticity equations to fit the IGABEM will be explained. The main problems that arises when doing this will also be commented. In the end, the MATLAB code, analogous to the previous one, will be resumed with its main functionalities presented. Again, a flux diagram is key to the overall comprehension.

In Chapter 6 a benchmark example will be shown to compare the accuracy of the different methods. It is a pipe under internal pressure to serve as validation. Next to it a different problem, this time a thin plate with a hole, is solved to test the program in different conditions.

Finally, a summary of the presented work is offered, summing up the whole process as well as a brief overall view of the carried out procedure. It is accompanied by a list of possible future works that would improve and extend the validity of this method.

2 Anisotropic Elasticity

Since this work will study the behaviour of anisotropic materials, whose properties are usually less known than the isotropic ones, it is convenient to begin by stating the most important equations that rule this behaviour. This topic is discussed with greater detail by F.París, J. Cañas, J.C Marín and A.Barroso [6], or by T.C.T. Ting in [20].

2.1 Hooke's Law

The most generic relationship that can be established for a lineal elastic material in 3-D states that:

$$\sigma_{ij} = C_{ijkl}\varepsilon_{kl} \qquad i,j,k,l = 1,2,3 \tag{2.1}$$

where repetition of an index implies a summation in that index, σ is the stress tensor, ε means strain and **C** is the constitutive or elasticity tensor. Even though equation (2.1) establishes 81 different elastic constants, the restrictions about symmetry in the stress tensor can reduce them to 21, for the following properties must be checked:

$$C_{ijkl} = C_{jikl} = C_{ijlk} = C_{klij} \tag{2.2}$$

In order to simplify the notation, the following subscripts will be used:

$$ij \rightarrow \alpha \qquad where \begin{cases} \alpha = i & \text{if i=j} \\ \alpha = 9 - i - j & \text{if i}\neq\text{j} \end{cases} \tag{2.3}$$

With this in mind the constitutive relations (2.1) can be represented as a matrix:

$$\begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{23} \\ \sigma_{13} \\ \sigma_{12} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\ & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} \\ & & C_{33} & C_{34} & C_{35} & C_{36} \\ & & & C_{44} & C_{45} & C_{46} \\ & \text{sym.} & & & C_{55} & C_{56} \\ & & & & & C_{66} \end{bmatrix} \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ \gamma_{23} \\ \gamma_{13} \\ \gamma_{12} \end{bmatrix} \tag{2.4}$$

where $\gamma_{ij} = 2\varepsilon_{ij}$.
The inverse relationship to equation 2.4 will be referred as:

$$\begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ \gamma_{23} \\ \gamma_{13} \\ \gamma_{12} \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} & S_{13} & S_{14} & S_{15} & S_{16} \\ & S_{22} & S_{23} & S_{24} & S_{25} & S_{26} \\ & & S_{33} & S_{34} & S_{35} & S_{36} \\ & & & S_{44} & S_{45} & S_{46} \\ & \text{sym.} & & & S_{55} & S_{56} \\ & & & & & S_{66} \end{bmatrix} \begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{23} \\ \sigma_{13} \\ \sigma_{12} \end{bmatrix} \tag{2.5}$$

where **S** is known as the compliance tensor.

If the material has one plane of elastic symmetry, it is called monoclinic and equation 2.4 can be written with 13 constants:

$$\begin{bmatrix}\sigma_{11}\\ \sigma_{22}\\ \sigma_{33}\\ \sigma_{23}\\ \sigma_{13}\\ \sigma_{12}\end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & 0 & 0 & C_{16} \\ & C_{22} & C_{23} & 0 & 0 & C_{26} \\ & & C_{33} & 0 & 0 & C_{36} \\ & & & C_{44} & C_{45} & 0 \\ & \text{sym.} & & & C_{55} & 0 \\ & & & & & C_{66} \end{bmatrix} \begin{bmatrix}\varepsilon_{11}\\ \varepsilon_{22}\\ \varepsilon_{33}\\ \gamma_{23}\\ \gamma_{13}\\ \gamma_{12}\end{bmatrix} \tag{2.6}$$

In 2.6 the plane $x_1 - x_2$ has been set as the plane of symmetry.

If the material has 3 planes of elastic symmetry, it is called orthotropic and its number of independent constants can be further reduced to 9:

$$\begin{bmatrix}\sigma_{11}\\ \sigma_{22}\\ \sigma_{33}\\ \sigma_{23}\\ \sigma_{13}\\ \sigma_{12}\end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & 0 & 0 & 0 \\ & C_{22} & C_{23} & 0 & 0 & 0 \\ & & C_{33} & 0 & 0 & 0 \\ & & & C_{44} & 0 & 0 \\ & \text{sym.} & & & C_{55} & 0 \\ & & & & & C_{66} \end{bmatrix} \begin{bmatrix}\varepsilon_{11}\\ \varepsilon_{22}\\ \varepsilon_{33}\\ \gamma_{23}\\ \gamma_{13}\\ \gamma_{12}\end{bmatrix} \tag{2.7}$$

Finally, in an isotropic material (where all planes are of symmetry) only 2 constants will remain:

$$\begin{bmatrix}\sigma_{11}\\ \sigma_{22}\\ \sigma_{33}\\ \sigma_{23}\\ \sigma_{13}\\ \sigma_{12}\end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{12} & 0 & 0 & 0 \\ & C_{11} & C_{12} & 0 & 0 & 0 \\ & & C_{11} & 0 & 0 & 0 \\ & & & \frac{C_{11}-C_{12}}{2} & 0 & 0 \\ & \text{sym.} & & & \frac{C_{11}-C_{12}}{2} & 0 \\ & & & & & \frac{C_{11}-C_{12}}{2} \end{bmatrix} \begin{bmatrix}\varepsilon_{11}\\ \varepsilon_{22}\\ \varepsilon_{33}\\ \gamma_{23}\\ \gamma_{13}\\ \gamma_{12}\end{bmatrix} \tag{2.8}$$

2.2 Plane and antiplane problems

In plane elasticity, if the material is monoclinic (or one of the cases in equations 2.7 and 2.8), the problem can be split in two separate sets of equations known as the plane and antiplane problems, that can be solved separately. The plane problem is:

$$\begin{bmatrix}\sigma_{11}\\ \sigma_{22}\\ \sigma_{12}\end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{16} \\ & C_{22} & C_{26} \\ \text{sym.} & & C_{66} \end{bmatrix} \begin{bmatrix}\varepsilon_{11}\\ \varepsilon_{22}\\ \gamma_{12}\end{bmatrix} \tag{2.9}$$

With

$$\left.\begin{aligned} \varepsilon_{33} &= 0 \\ \sigma_{33} &= C_{13}\varepsilon_{11} + C_{23}\varepsilon_{22} + C_{36}\gamma_{12} \end{aligned}\right\} = \text{for plane strain problems} \tag{2.10}$$

$$\left.\begin{aligned} \varepsilon_{33} &= -\frac{C_{13}\varepsilon_{11} + C_{23}\varepsilon_{22} + C_{36}\gamma_{12}}{C_{33}} \\ \sigma_{33} &= 0 \end{aligned}\right\} = \text{for plane stress problems} \tag{2.11}$$

As for the antiplane problem:

$$\begin{bmatrix}\sigma_{23}\\ \sigma_{13}\end{bmatrix} = \begin{bmatrix} C_{44} & C_{45} \\ C_{45} & C_{55} \end{bmatrix} \begin{bmatrix}\gamma_{23}\\ \gamma_{13}\end{bmatrix} \tag{2.12}$$

3 Boundary Integral Equations

This chapter will cover the theory behind both BEM and IGABEM methods, presenting the equations that will be particularized and used later.

3.1 Somigliana's identity

The starting point for the mathematical proof of this identity as detailed on the work of Brebbia and Dominguez in [5]) will be the equilibrium equations of elasticity for a body of domain Ω and boundary σ. Since this work, for simplicity, disregards body forces, such expression is:

$$\sigma_{kj,j} = 0 \qquad \text{in } \Omega \tag{3.1}$$

In order to minimize (3.1), it can be weighted by a displacement type function u_k^*. Therefore:

$$\int_\Omega \sigma_{kj,j} u_k^* d\Omega = 0 \tag{3.2}$$

By carrying out an integration by parts twice in the first term of equation (3.2), it is modified to:

$$\int_\Omega \sigma_{kj,j}^* u_k d\Omega + \int_\Gamma p_k u_k^* d\Gamma = \int_\Gamma p_k^* u_k d\Gamma \tag{3.3}$$

where σ_{kj}^* and p_k^* are the stresses and tensions related to u_k^*. Now, if the introduced u_k^* function corresponds to the displacement created by and unit force in direction l applied at point y (this function is called fundamental solution), the domain integral only is non null in that particular point, that is:

$$\int_\Omega \sigma_{kj,j}^* u_k d\Omega = \int_\Omega \sigma_{lj,j}^* u_l d\Omega = u_l^y \tag{3.4}$$

As stated before, u^* and p^* are called fundamental solutions and will be explained in more detail in the next chapter. The important point here is that among all the arbitrary functions that could have been chosen, u^* is picked so that the integral in 3.4 becomes a punctual value and p^* steams from u^*. Now (3.4) can be substituted into (3.3) by taking each of the separate components of the displacement, and the Somigliana's identity is reached:

$$u_l^y + \int_\Gamma p_{lk}^* u_k d\Gamma = \int_\Gamma u_{lk}^* p_k d\Gamma \tag{3.5}$$

This equation is the basis for the boundary element method. It allows us to work out the displacement in any point of the domain provided the tractions and displacements at the boundaries are known. It is true for any point of the solid, be it internal or on the boundary itself.

3.2 Fundamental solutions

In order to apply the above equation (3.5), it is necessary to know the expressions of p_{lk}^* and u_{lk}^*. If the x_1x_2-plane is taken as the plane of study, p_{lk}^* and u_{lk}^* are the tractions and displacements, respectively, created

at point $\mathbf{x}$ (x_1, x_2) when a constant unit load along the x_3-axis in the k direction is applied at point $\mathbf{y}$ (y_1, y_2). The $\mathbf{x}$ point is called observation point and $\mathbf{y}$ is referred as the collocation point. For an anisotropic bidimensional solid in plane strain and following the nomenclature used by García-Sánchez in [7], these expressions are (where l is the direction of the component of the solution):

$$u^*_{lk} = -\frac{1}{\pi} Re(A_{km} Q_{ml} ln(z_m - z^0_m) \tag{3.6}$$

$$p^*_{lk} = \frac{1}{\pi} Re(L_{km} Q_{ml} \frac{\mu_m n_1 - n_2}{z_m - z^0_m}) \tag{3.7}$$

Where:

- The indices l,k,m=1,2 if only the plane problem is being solved (when the material properties allow it, see section 2.2). Otherwise, l,k,m=1,2,3.
- Re(x) stands for the real part of x.
- n_k is the k component of the exterior normal to the boundary Γ at point $\mathbf{x}$.
- z_m is the transformation to the complex plane of the observation point:

$$z_m = x_1 + \mu_m x_2 \tag{3.8}$$

- z^0_m is the transformation to the complex plane of the collocation point:

$$z^0_m = y_1 + \mu_m y_2 \tag{3.9}$$

- μ_m are the complex roots with positive imaginary part of the equation:

$$det[C_{2ij2}\mu^2_m + (C_{1ij2} + C_{2ij1})\mu_m + C_{1ij1}] = 0 \tag{3.10}$$

- The matrix $\mathbf{A}$ is made, by columns, from the eigenvectors resulting from solving the characteristic equation of the material:

$$[C_{2ij2}\mu^2_m + (C_{1ij2} + C_{2ij1})\mu_m + C_{1ij1}]A_{jm} = 0 \qquad \text{(no sum in m)} \tag{3.11}$$

  where the $C_{\alpha i j \beta}$ sub-matrices follow the expressions in 2.3.
- The matrix $\mathbf{Q}$ is defined as:

$$\mathbf{Q} = \mathbf{A^{-1}}(\mathbf{B^{-1}} + \overline{\mathbf{B}}^{-1})^{-1} \tag{3.12}$$

- The $\mathbf{B}$ matrix is defined as:

$$\mathbf{B} = i\mathbf{A}\mathbf{L}^{-1} \tag{3.13}$$

- $\bar{\mathbf{B}}$ stands for the complex conjugate matrix of $\mathbf{B}$.
- The matrix $\mathbf{L}$ is made, by columns:

$$L_{im} = (\mu_m C_{2ij2} + C_{2ij1})A_{jm} = -\frac{1}{\mu_m}(C_{1ij1} + \mu_m C_{1ij2})A_{jm} \qquad \text{(no sum in m)} \tag{3.14}$$

It is worth noting that expressions 3.10 to 3.14 depend entirely on the properties of the material, not the geometry, and are constants and not variables.

3.2.1 Isotropic materials as a degenerate case

T.C.T Ting explores ([20]) how the constants in equations 3.10 to 3.14 are depending on the material. It is interesting to point out that μ_m cannot be real, and that they will come in conjugate pairs. In the particular case of an isotropic material, all roots will be equal and will take the value of i. This will cause the columns of the $\mathbf{B}$ matrix to be linearly dependent and its inversion in 3.12 problematic. In the MATLAB program used in this work, the determinant of $\mathbf{B}$ is, albeit very small, not exactly zero, which allows it to work for

isotropic materials despite being a poorly conditioned problem. The results are acceptably accurate despite this, but it would be advisable to use the fundamental solution for the isotropic case for greater accuracy. For the sake of completeness, they will be covered in this section.

The expression of the fundamental solutions (particularized for a 2D isotropic plane strain problem) are:

$$u^*_{lk} = \frac{1}{8\pi\mu(1-\nu)}[(3-4\nu)ln\frac{1}{r}\delta_{lk} + r_{,l}r_{,k}] \tag{3.15}$$

$$p^*_{lk} = -\frac{1}{4\pi(1-\nu)r}[r_{,n}((1-2\nu)\delta_{lk} + 2r_{,l}r_{,k}) + (1-2\nu)(n_l r_{,k} - n_k r_{,l})] \qquad l,k = 1,2 \tag{3.16}$$

where r is the distance between the collocation and observation points, μ is the Shear Modulus, ν is the Poisson's ratio, δ_{lk} is the Kronecker delta and $r_{,n}$ is the derivative of r with respect to the normal, that is:

$$r_{,n} = \frac{\partial r}{\partial x_k}n_k + \frac{\partial r}{\partial x_l}n_l \tag{3.17}$$

3.3 Boundary Integral Equations

3.3.1 Boundary Points

As expressed in the first section, it is first needed to solve the problem in the boundary to solve the whole domain. Somigliana's identity is valid for any collocation point, including the boundary. A problem arises, nonetheless, when this happens, since r causes a singularity when the collocation and integration points match. In order to avoid this singularity, the integration can be carried out around the collocation point, as shown in Figure 3.1. (where Γ_ε is the part of the boundary where the collocation point lies).

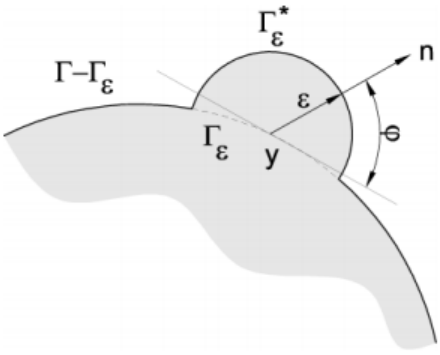


Figure 3.1 Integration around the singularity. Figure taken from [17].

The equation (3.5) would transform into:

$$u^y_l + \lim_{\varepsilon\to 0}\int_{\Gamma-\Gamma_\varepsilon} p^*_{lk}u_k d\Gamma + \lim_{\varepsilon\to 0}\int_{\Gamma_\varepsilon} p^*_{lk}u_l d\Gamma = \lim_{\varepsilon\to 0}\int_{\Gamma-\Gamma_\varepsilon} u^*_{lk}p_k d\Gamma + \lim_{\varepsilon\to 0}\int_{\Gamma_\varepsilon} u^*_{lk}p_k d\Gamma \tag{3.18}$$

Only the third and fifth terms bring further trouble. The second and fourth are integrations in the sense of Cauchy principal value. In the fifth, as the limit is approached, p_k will become its punctual value and can be placed out of the integral.

$$\lim_{\varepsilon\to 0} p_k \int_{\Gamma_\varepsilon} u^*_{lk} d\Gamma \tag{3.19}$$

u^*_{lk} will produce a term of order $ln(1/\varepsilon)$ whereas $d\Gamma$ is a term of order ε, therefore the value of the limit tends to 0. On the third one the fundamental equation is this time of order ε, the same as the length differential, so

a non zero coefficient will appear.

$$\lim_{\varepsilon \to 0} u_k \int_{\Gamma_\varepsilon} p^*_{lk} d\Gamma = c^*_{lk} u_k \tag{3.20}$$

This term c^*_{lk} will depend on the shape of the surface where the integration is being carried out as well as the properties of the material. The calculation of these values may prove difficult (especially in 3D) but, as will be seen, it can be avoided. When c^*_{lk} is moved to the other side of the equation and summed to the displacement term that is there, the result is c_{lk} and is typically referred as free term coefficient or jump term coefficient.

In conclusion, the equation (3.5) has been transformed into:

$$c^y_{lk} u^y_k + \int_\Gamma p^*_{lk} u_k d\Gamma = \int_\Gamma u^*_{lk} p_k d\Gamma \tag{3.21}$$

In this expression the integrals are in the sense of Cauchy principal value. This equation will be the central axis of this work and the key to solve firstly the boundary, and secondly the domain.

3.3.2 Internal Points

When the collocation point lays inside the solid, the singularity no longer takes place. Therefore equation (3.21) can be used with $c_{lk} = 1$ which equals the original (3.5). Known the boundary values, the displacement of any internal point can be easily found. For the computation of stresses, equation 3.21 can be differentiated and the cinematic relationship applied, arriving to an integral equation that allows the computation of stresses:

$$\sigma^y_{sl} + \int_\Gamma s^*_{slk} u_k d\Gamma = \int_\Gamma d^*_{slk} p_k d\Gamma \tag{3.22}$$

where:

$$s^*_{slk} = C_{sljm} p^*_{jk,m} \tag{3.23}$$

$$d^*_{slk} = C_{sljm} u^*_{jk,m} \tag{3.24}$$

The indices s,l,k,m follow the same rule as in section 3.2.

Isotropic materials as a degenerate case

If the same differentiation is made taking the isotropic fundamental solution, the equation equivalent to 3.22 can be written with a slightly different nomenclature, since the derivatives of the fundamental solutions, being simpler, are already included:

$$\sigma^y_{ij} + \int_\Gamma s^*_{kij} u_k d\Gamma = \int_\Gamma d^*_{kij} p_k d\Gamma \tag{3.25}$$

where d^*_{kij} and s^*_{kij} are:

$$d^*_{kij} = \frac{1}{4\pi(1-\nu)r}\Big((1-2\nu)(\delta_{kij} r_{,j} + \delta_{kj} r_{,i} - \delta_{ij} r_{,k}) + 2r_{,i} r_{,j} r_{,k}\Big) \tag{3.26}$$

$$\begin{aligned} s^*_{kij} = \frac{\mu}{2\pi(1-\nu)r^2}\Big(2\frac{\partial r}{\partial n}[(1-2\nu)\delta_{ij} r_{,k} + \nu(\delta_{ik} r_{,j} + \delta_{jk} r_{,i}) - 4r_{,i} r_{,j} r_{,k}] + \\ +2\nu(n_i r_{,j} r_{,k} + n_j r_{,i} r_{,k}) + (1-2\nu)(2n_k r_{,i} r_{,j} + n_j \delta_{ik} + n_i \delta_{jk}) - (1-4\nu) n_k \delta_{ij}\Big) \end{aligned} \tag{3.27}$$

4 Boundary Element Method

In this chapter, the traditional BEM will be briefly discussed. It will show how the equations are set and the most important details about their resolution. When the discretization is discussed, it will be generally assumed that only the plane problem is being solved for simplicity regarding indices and sizes. For the most generically anisotropic case, specifics will be mentioned when needed.

4.1 Boundary Elements Discretization

In order to solve the boundary, some assumptions must be made. It is going to be split in a number NE of elements with a quantity n of nodes. Since quadratic elements will be used in this work, each element will be formed by three nodes, one in its middle and one in each end as can be seen in Figure 4.1 . Therefore, $N = 2 * NE$.

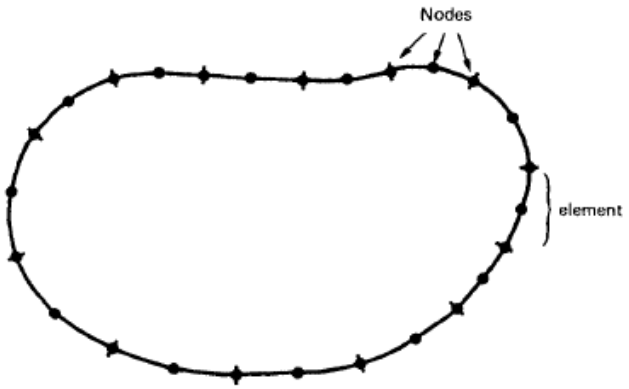


Figure 4.1 Discretization with quadratic elements. Figure taken from [5].

The position, displacement and tractions of any point along an element will be presumed as a combination of the nodal values. This way, shape functions $\phi(\xi)$ are introduced, so that:

$$\mathbf{x}(\xi) = \phi_1(\xi)\mathbf{x}^1 + \phi_2(\xi)\mathbf{x}^2 + \phi_3(\xi)\mathbf{x}^3 \tag{4.1}$$

$$\mathbf{u}(\xi) = \phi_1(\xi)\mathbf{u}^1 + \phi_2(\xi)\mathbf{u}^2 + \phi_3(\xi)\mathbf{u}^3 \tag{4.2}$$

$$\mathbf{p}(\xi) = \phi_1(\xi)\mathbf{p}^1 + \phi_2(\xi)\mathbf{p}^2 + \phi_3(\xi)\mathbf{p}^3, \tag{4.3}$$

where $\mathbf{x^i}$, $\mathbf{u^i}$ and $\mathbf{p^i}$ are the geometry, displacements and tractions, respectively, at node i. ξ is a parametric coordinate which goes over the element varying from -1 to 1, this is, the first node will be placed at $\xi = -1$,

the second at $\xi = 0$ and the third at $\xi = 1$. The expressions of the shape functions are as follows:

$$\phi_1(\xi) = \frac{1}{2}\xi(\xi - 1) \tag{4.4}$$

$$\phi_2(\xi) = (1 - \xi^2) \tag{4.5}$$

$$\phi_3(\xi) = \frac{1}{2}\xi(\xi + 1) \tag{4.6}$$

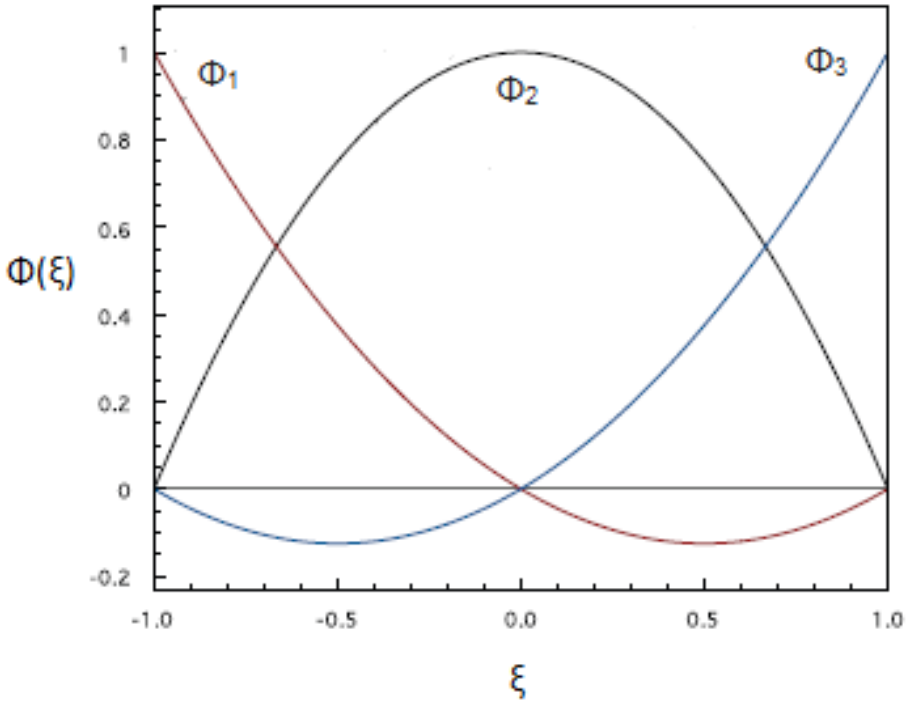


Figure 4.2 Quadratic shape functions.

A remarkable property is that all the functions are zero at any node different from the one they are related to. The meaning of this is simply that it is assumed that the function value at a node is precisely the value of the function at such node, with no influence of other nodes. A representation of these three functions can be seen at Figure 4.2

Additionally, while the displacement value is unique for a node, its traction is not since it is related to a normal and any node in a corner will have two. Therefore, there will be one traction per node per element, as Figure 4.3 depicts, where only one component of each vector has been drawn for simplicity (all of them in the first direction).

It can be seen that both t_1^{31} (belonging to the third node of the first element) and t_1^{12} (belonging to the first node of the second element) correspond to the same node, and yet they will be different in general.

Having already stated this discretization, Somigliana's identity (3.21) will transform into:

$$c_{lk}(y)u_k^y + \sum_{e=1}^{NE}\sum_{j=1}^{3}\left[\int_{-1}^{1} p_{lk}^*(y,\xi)\phi_j(\xi)|J|(\xi)d\xi\right]u_j^{je} =$$

$$= \sum_{e=1}^{NE}\sum_{j=1}^{3}\left[\int_{-1}^{1} u_{lk}^*(y,\xi)\phi_j(\xi)|J|(\xi)d\xi\right]p_k^{je} \quad i,j = 1,2 \tag{4.7}$$

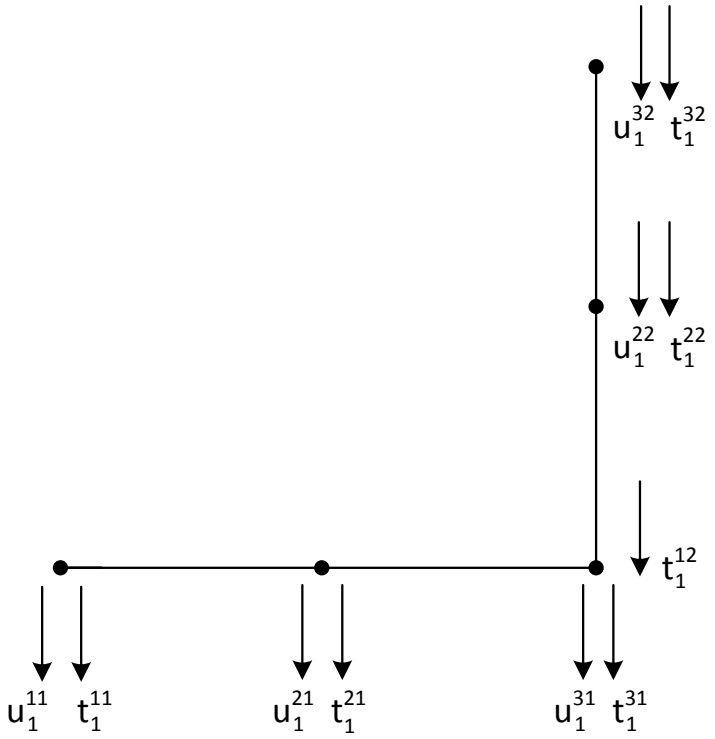


Figure 4.3 Variables associated to nodes.

where u_j^y is the displacement in direction j of the collocation point y and p_k^{je} means the traction in direction k at node j of the element e. $|J|$ is the Jacobian of the transformation, that is:

$$|J| = \frac{d\Gamma}{d\xi} = \sqrt{\left(\frac{dx_1}{d\xi}\right)^2 + \left(\frac{dx_2}{d\xi}\right)^2} \tag{4.8}$$

This expression can be computed using equation (4.1).

The results of these integrations will be stored in two matrices called $\hat{\mathbf{H}}$ and $\mathbf{G}$, each one containing the integrals of $\mathbf{p}^*$ and $\mathbf{u}^*$ respectively. Then, c_{lk} will be added to the proper position of $\hat{\mathbf{H}}$ to form the matrix $\mathbf{H}$.

After positioning the unit force in the N different nodes, a system of equations will be built:

$$\mathbf{Hu} = \mathbf{Gp} \tag{4.9}$$

in which $\mathbf{u}$ and $\mathbf{t}$ are partially known. These matrices can be rearranged in regard to the boundary conditions to form an equation of the type $\mathbf{Ax} = \mathbf{B}$ which can be subsequently solved.

The $\mathbf{H}$ matrix will be square of size $2N \times 2N$ and $\mathbf{G}$ will be rectangular of size $2N \times 3N$. In both of them, the row number signs the collocation point and the column, the integration one.

4.2 Numerical implementation

The code used for these results is an adaptation of the one provided by [4].

4.2.1 Rigid body considerations

As can be seen in equation (4.7) the c_{lk} affects to the diagonal terms of matrix $\mathbf{H}$ because it only applies when the collocation and integration points coincide. Whether diagonal is said, it must be understood in an

extended sense. For example, in the first row it will appear in both the first and second columns, being both related to the collocation point ($\mathbf{u_1^1}$ and $\mathbf{u_2^1}$ respectively).
In case the solid suffers a unit rigid body displacement in a particular direction, that is, $\mathbf{p} = \mathbf{0}$, $\mathbf{u_1} = \mathbf{1}$ and $\mathbf{u_2} = \mathbf{0}$; the equation must still be valid. This forces the matrix $\mathbf{H}$ to fulfil the next property:

$$H_{ii} = -\sum_{j=1}^{n/2} H_{i,2j-1} \qquad \text{for} \quad 2j-1 \neq i \tag{4.10}$$

Since the same reasoning can be made for a displacement in the other direction, the former equation is valid for even indexes too. This is, not only do the rows of $\mathbf{H}$ have to sum zero, but the even and odd positions have to sum zero as well. This way, the diagonal terms of the matrix can be found without the need of using the terms c_{lk}.

4.2.2 Singular integrals in matrix H

Although these integrals are considered in the sense of Cauchy principal values, discontinuities that emerge when the collocation point lays in the same element being integrated and additional precautions must be taken. In $\mathbf{H}$, nevertheless, one property comes handy.
As stated previously, shape functions take the value 0 at any node other that the one they are attached to. Equations 3.7 and 4.6 have discontinuities of the same order, which means that $\lim_{x\to y}(p^*_{lk}\phi_j)$ is finite for any k different from the one where the collocation point is. The particular value of the limit depends on the case. When k takes the value of the collocation point, the integration does not need to be carried, for these are the terms obtained by the rigid body considerations.

4.2.3 Traction discontinuities at corner points

Another case that deserves special attention are corners with traction discontinuities, and is related to the number of unknown variables. The possible boundary conditions at the corner are as follow (considering only one direction for simplicity, which can be easily applied to the other direction):

- If the tractions before and after the corner node are given, the displacement is the only unknown variable. Therefore, there is one equation for one unknown and there is not a problem.
- If the displacement and one of the tractions are given (either before or after), the unknown is the remaining traction. Again, there is not a problem since there is only one unknown.
- However, if the boundary conditions in both elements are given in displacement, there are two unknowns to solve, the two tractions. The second boundary condition is giving redundant information, which makes this case unlike the previous two. If this happens, additional action must be taken.

The easiest way to solve this problem is to duplicate the node, creating a very small element between the two duplicates. Although this modifies the geometry of the problem, results are accurate as long as the distance is small enough. A graphical example is shown in Figure 4.4.

4.2.4 Singular integrals in matrix G

While most integrals are computed with the use of a special Gauss quadrature with ten terms, in the singular integrals an additional step is required in order to use a special integration formula. The idea is, by means of a change of variable, to split the integral into one with a singularity and another without. The different cases (depicted in Figure 4.5) will be analyzed next.

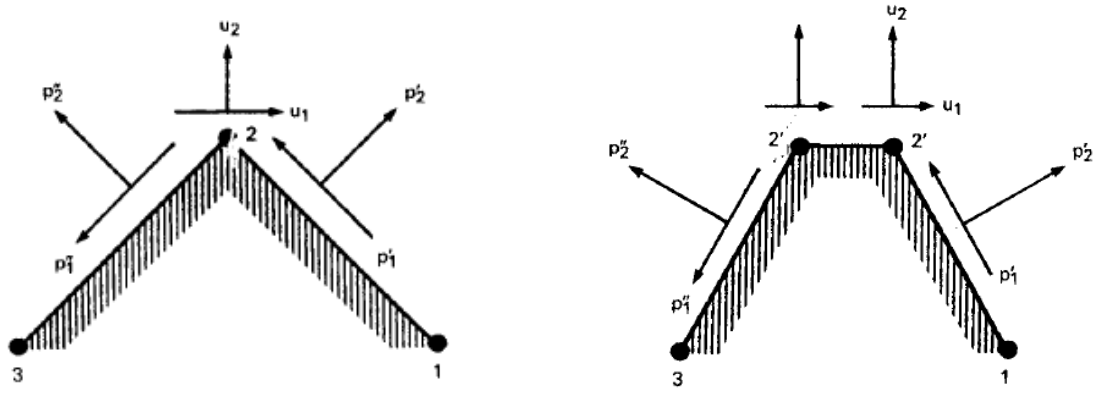
Collocation point at node 1

Here the new variable will be:

$$\eta = \frac{\xi + 1}{2} \tag{4.11}$$

And the following abbreviation will be used:

$$\chi_m = z_m - z_m^0 \tag{4.12}$$



(a) Original node. **(b)** Duplicated node with a small element.

Figure 4.4 Inserting element at a corner. Figure taken from [5].

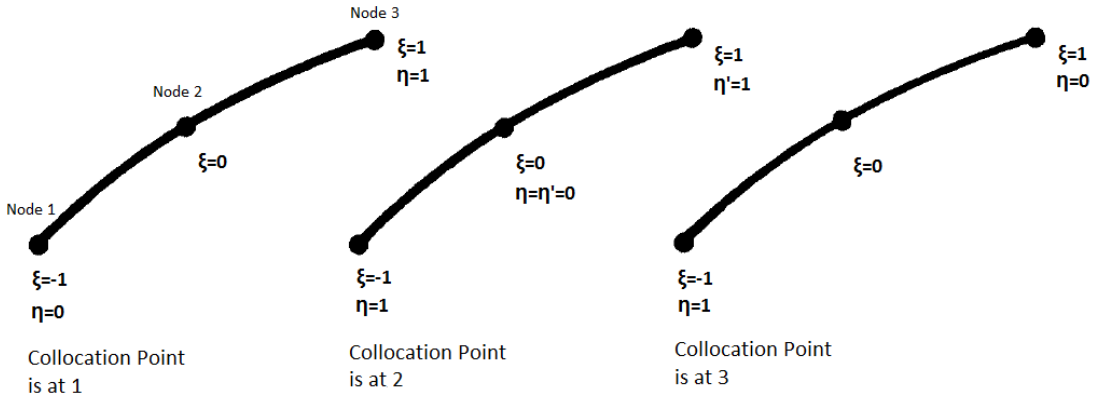


Figure 4.5 Coordinate systems for numerical integration. Figure taken from [5].

The non-problematic part of u^*_{ij} will be omitted for simplicity, and it will be shown only for one m (the integral of the sum can be divided in the sum of the integrals, and the transformation done to all parts). The transformation is:

$$\int_{-1}^{1} \ln\left(\frac{1}{\chi_m(\xi)}\right)\phi_k(\xi)|J|(\xi)d\xi = \int_{0}^{1} \ln\left(\frac{\eta}{\chi_m(\eta)\eta}\right)\phi_k(\eta)|J|(\eta)2d\eta =$$

$$= \int_{0}^{1} \ln\left(\frac{\eta}{\chi_m(\eta)}\right)\phi_k(\eta)|J|(\eta)2d\eta + \int_{0}^{1} \ln\left(\frac{1}{\eta}\right)\phi_k(\eta)|J|(\eta)2d\eta \tag{4.13}$$

It must be noticed that, in the last expression the first addend does not present a singularity since the limit of the logarithm near $\chi_m = 0$ is of a lower order than the shape function. The second addend (and all other singular integrals presented further in this subsection) will be computed using an standard logarithmic Gauss quadrature, which states that:

$$\int_{0}^{1} \ln\left(\frac{1}{\eta}\right) f(\eta) \simeq \sum_{i=1}^{n} w_i f(\eta_i) \tag{4.14}$$

Collocation point at node 2

In this one, the integral will be firstly split in two halves, which will be dealt separately:

$$\int_{-1}^{1} \ln\left(\frac{1}{\chi_m(\xi)}\right)\phi_k(\xi)|J|(\xi)d\xi = \int_{-1}^{0} \ln\left(\frac{1}{\chi_m(\xi)}\right)\phi_k(\xi)|J|(\xi)d\xi + \int_{0}^{1} \ln\left(\frac{1}{\chi_m(\xi)}\right)\phi_k(\xi)|J|(\xi)d\xi \tag{4.15}$$

In the first part, $\eta = -\xi$ will be used; in the second, the variable will be $\eta' = \xi$:

$$\int_0^1 \ln\left(\frac{1}{\chi_m(\xi)}\right)\phi_k(\xi)|J|(\xi)d\xi + \int_0^1 \ln\left(\frac{1}{\chi_m(\xi)}\right)\phi_k(\xi)|J|(\xi)d\xi =$$

$$= \int_0^1 \ln\left(\frac{\eta}{\chi_m(\eta)\eta}\right)\phi_k(\eta)|J|(\eta)d\eta + \int_0^1 \ln\left(\frac{\eta'}{\chi_m(\eta')\eta'}\right)\phi_k(\eta')|J|(\eta)d\eta' =$$

$$= \int_0^1 \ln\left(\frac{\eta}{\chi_m(\eta)}\right)\phi_k(\eta)|J|(\eta)d\eta + \int_0^1 \ln\left(\frac{1}{\eta}\right)\phi_k(\eta)|J|(\eta)d\eta +$$

$$+ \int_0^1 \ln\left(\frac{\eta'}{\chi_m(\eta')}\right)\phi_k(\eta')|J|(\eta)d\eta' + \int_0^1 \ln\left(\frac{1}{\eta'}\right)\phi_k(\eta')|J|(\eta)d\eta' \tag{4.16}$$

In this last expression, terms one and three do not suffer a singularity; the limits of both the numerator and denominator inside the logarithm near $\chi_m = 0$ are of the same order. The second and fourth ones use (4.14).

Collocation point at node 3

The reasoning in this case is almost identical to the first one, with the only difference being the variable:

$$\eta = \frac{\xi - 1}{2} \tag{4.17}$$

Apart from that, all the steps are completely analogous.

4.2.5 Code organization

Here the overall function of the different subprograms will be briefly commented. The next figure explains the calling order and the recurrences made.

ELQUABEANI

This is the main program. It calls the others in duly order (see Figure 4.6) and solves the system $Ax = B$. Figure 4.6 shows the main idea of the method; after having ELQUABEANI start the whole process and INPUTEQANI read the data, GHMATEQANI sets all the nodes as collocation points in order and carry out the integrations along each element. Whenever it needs to integrate over an element that contains the collocation point, it calls both LOCINEQANI (a subprogram prepared to deal with singularities) and EXTINEQUANI (the basic integration program not capable to deal with singularities). For elements that are further from the collocation point, EXTIMEQANI suffices. After all the nodes have been set as collocation points, the resulting system of equation is solved.

For internal points ELQUABEANI calls INTEREQANI, which sets each internal point as collocation points and integrates along the boundary again, using EXTINEQANI and SIGMAEQANI.

This is based on the original BEM program by [4].

INPUTEQANI

This subroutine reads the input data from the designed .txt. This data contains the number of boundaries, the number of elements, the number of internal points to solve, the behaviour matrix , the coordinates of both boundary nodes and internal points and boundary conditions.
One important detail to note is that external boundaries data must be written counter-clockwise, while internal ones must be introduced clockwise. This is so the normal of the surfaces aim in the correct direction. All equations are set to solve plane strain problems, since the fundamental solution is valid for this kind. If a plane stress problem is to be solved, the user must introduce modified values of the behaviour matrix:

$$C'_{ij} = C_{ij} - C_{i3}C_{3j}/C_{33} \tag{4.18}$$

A vector called KODE informs about each node per element (that is, frontier nodes may be repeated or have different values), taking the value 0 if the displacement is known or 1 if the traction is. DFI takes the numerical values of the KODE data. All other variables are easily understood in the code.

GHMATEQANI

This subprogram computes the matrices $\mathbf{H}$ and $\mathbf{G}$. For each collocation point, it goes over every element calling EXTINEQANI or LOCINEQANI, depending on the situation. If no singularity is presented along the

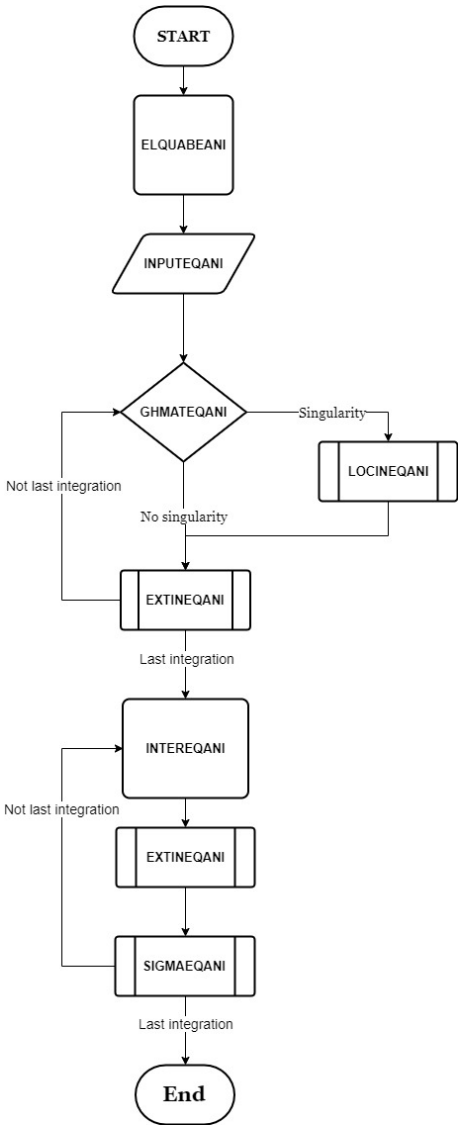


Figure 4.6 Flow diagram BEM.

selected element, EXTINEQANI computes all terms. If, however, the integral is singular, LOCINEQANI will be called to compute the **G** terms, while EXTINEQANI figures the **H** terms out.
After this, the rigid body consideration is used to overwrite the terms EXTINEQANI miscalculated.
Finally, it changes the columns from **H** and **G** attending to the boundary conditions, leading to a system $Ax = B$ ready to be solved by ELQUABEANI.

The number of integrations that are carried out are one per node per element, which is the number of times the loop in the flow diagram is repeated. After the last integration, the program can move on.

EXTINEQANI

This one computes the twelve terms (two components per node, with two direction of load application) of each matrix related to the present element if no singularities arise, and only the **H** terms when they do. This is achieved by a standard ten points Gauss quadrature.

LOCINEQANI

This subprogram computes the twelve terms of matrix **G** by means of a special Gauss quadrature, as was already detailed in 4.2.4.

INTEREQANI

This program first sorts the solution obtained by tractions and displacements. Then it uses the previous results to compute the displacements and tensions in internal points with the use of (3.21) and (3.25). It calls

EXTINEQANI for the new integrals (no singularities arise this time since the collocation point is inside the solid, not in the boundary) and SIGMAEQANI.

The number of integrations that are carried out are one per internal point per element, which is the number of times the loop in the flow diagram is repeated. After the last integration, the program can move on.

SIGMAEQANI

This routine gives back the needed terms shown in 3.3.2. These integrals are also computed by means of a standard Gauss quadrature.

5 Isogeometric Boundary Element Method

In this chapter the differences and similarities between BEM and IGABEM will be explored. The initial problems raised at working with the IGABEM will be presented and their solution detailed.

As introduced in the abstract, NURBS are a special group of functions used by CAD programs to build the boundary of items being designed. The IGABEM has the goal of using these functions in substitution of the shape functions in 4.6 as well as replacing the nodes with a set of points that CAD software programs use. This would ease the step from CAD design to analysis. This apparently simple change has some implications that will be discussed in along the chapter.

5.1 Non-Uniform Rational B-Splines

Since NURBS are built from B-Splines, it makes quite sense to introduce them before any step ahead is given.

5.1.1 B-Splines

B-Splines are parametric functions that map the parameter space into the physical space. In this work η will be used for the parametric variable. This variable is somehow similar to the ξ used in traditional BEM to define the shape functions. The main difference is that η is not local to the elements.

Possibly the most important components of B-Splines are the knot vector, non-decreasing sequence of coordinates usually referred by $\Xi = (\eta_1,\eta_2,...\eta_{n+p+1})$ where p is the grade the polynomial order (p=2 from now on, for quadratic elements will be employed), n the number of basis functions used and η_i, the knots. This vector splits η into elements. Knot vectors are uniform if the knot are equally distributed in the parameter space, and non-uniform otherwise. They are open if the initial and final knots are repeated $p+1$ times. This means the splines will be interpolatory at the beginning and end of the curve, and is the most frequent kind of knot vector.

The repetition of knot values entails the decrease of continuity order at that point in the spline. In fact, N_{a_p} is $p-l$ times differentiable at a knot of multiplicity l.

Besides, it is worth noting that the knot vector is indifferent to its scale, that is, the vector (0,0,0,2,2,4,4,4) with $\eta \in [0,4)$ is equivalent to the vector (0,0,0,1,1,2,2,2) with $\eta \in [0,2)$. With this, B-Splines for grade p are defined as:

$$N_{a\_p}(\eta) = \frac{\eta - \eta_a}{\eta_{a+p} - \eta_a} N_{a\_p-1}(\eta) + \frac{\eta_{a+p+1} - \eta}{\eta_{a+p+1} - \eta_{a+1}} N_{a+1\_p-1}(\eta) \tag{5.1}$$

With the exception of grade 0 functions

$$N_{a\_0}(\eta) = \begin{cases} 1 \text{ if } \eta_a \leq \eta < \eta_{a+1} \\ 0 \text{ otherwise} \end{cases} \tag{5.2}$$

If equation (5.1) yields the quotient $\frac{0}{0}$, it is taken as zero by definition.
Figure 5.1 shows an example of B-splines that will be useful for highlighting some facts.

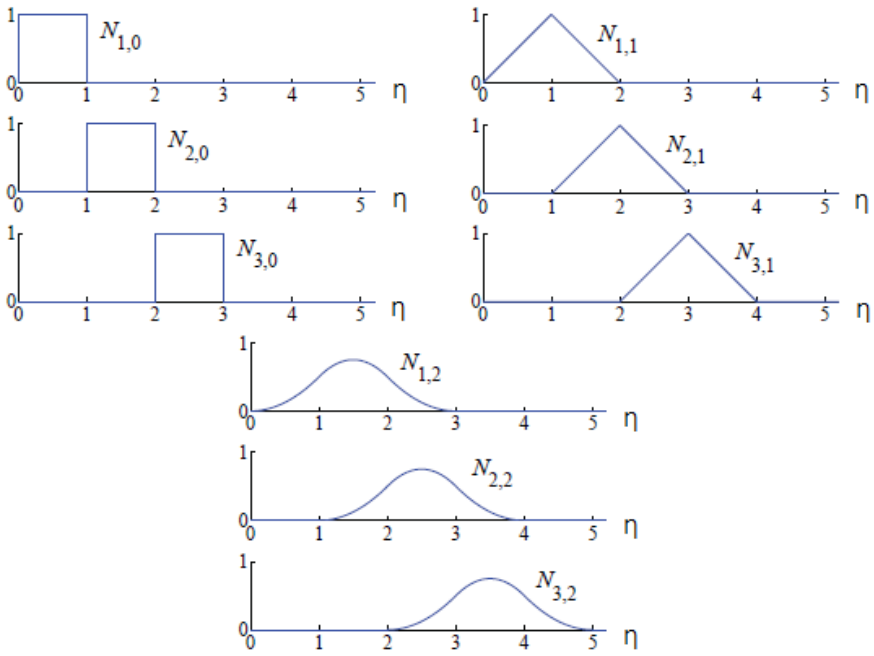


Figure 5.1 B-splines for the knot vector $\Xi = (0,1,2,3,4...)$. Figure taken from [3].

The first property to be noticed is that, since B-splines are a partition of the unity:

$$\sum_{k=1}^{n} N_{a\_p}(\eta) = 1; \qquad \forall \eta \tag{5.3}$$

Secondly, in any point, at most $p+1$ functions are different from zero. Finally, $N_{a\_p}(\eta) = 0$ out of the span $[\eta_a, \eta_{a+p+1})$.

5.1.2 Building NURBS

NURBS are built from B-Splines in the following way:

$$R_{a\_p}(\eta) = \frac{N_{a\_p}(\eta) w_a}{\sum_{i=1}^{n} N_{i\_p}(\eta) w_i}, \tag{5.4}$$

where w_a is the weight of the function a. It is immediately seen that, if all weights are unit, the NURBS will coincide with the B-splines because of equation (5.3) . Moreover, NURBS also fulfil such equation.

The step from B-Splines to NURBS is significant, for it allows to describe certain curves, such as ellipses or circumferences, in an exact way, which was impossible with polynomial functions.

With the help of this functions, the coordinates of the curve C described by η are defined as:

$$\mathbf{C}(\eta) = \sum_{a=1}^{n} R_{a\_p}(\eta) \mathbf{P^a}, \tag{5.5}$$

where $\mathbf{P}$ are the coordinates of the control points. These control points are similar to the traditional BEM nodes but, very importantly, they do not necessarily belong to the curve. If the first and last control points are coincident, the curve is closed.

Figure 5.2 presents an example of a curve built from its control points. The knot vector associated to it is $\Xi = \left(0,0,0,\frac{1}{9},\frac{1}{9},\frac{2}{9},\frac{1}{3},\frac{4}{9},\frac{5}{9},\frac{2}{3},\frac{7}{3},\frac{8}{9},\frac{8}{9},1,1,1\right)$

5.2 Boundary Element Discretization

In a manner similar to 4.1, the next approximations are obtained:

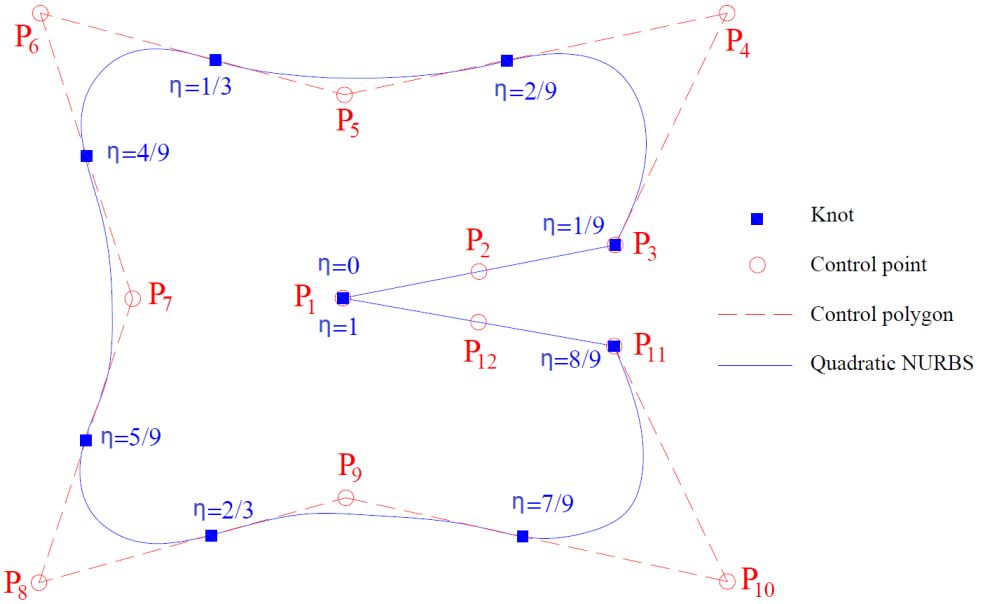


Figure 5.2 Geometry defined by NURBS. Figure taken from [13].

$$\mathbf{x}(\eta) = \sum_{a=1}^{n} R_{a\_p}(\eta)\mathbf{P^a} \tag{5.6}$$

$$\mathbf{u}(\eta) = \sum_{a=1}^{n} R_{a\_p}(\eta)\mathbf{d^a} \tag{5.7}$$

$$\mathbf{p}(\eta) = \sum_{a=1}^{n} R_{a\_p}(\eta)\mathbf{q^a} \tag{5.8}$$

The reason behind having expanded this relations is to draw attention to the **d** and **q** vectors. This are called the displacement and traction coefficients, respectively. The term coefficient is used because they do not represent the nodal points displacement or tractions even if such points were all in the boundary (it is worth repeating that the control points do not necessary belong to the curve so they can be placed inside the solid, outside of it or in the boundary). Furthermore, these vectors have no physical meaning.

The role of elements will be taken by the different knot spans, provided their length is longer that zero.

Analogously to in traditional BEM, the Somigliana's identity (3.21) can be made discreet as:

$$c_{lk}(y)\sum_{a=1}^{n} R_{a\_p} d_j^y + \sum_{e=1}^{ne}\sum_{a=1}^{n}\left[\int_{-1}^{1} p_{lk}^*(y,\xi)R_{a\_p}(\xi)|J|(\xi)|J_2|d\xi\right]d_k^{ae} =$$

$$= \sum_{e=1}^{ne}\sum_{a=1}^{n}\left[\int_{-1}^{1} u_{lk}^*(y,\xi)R_{a\_p}(\xi)|J|(\xi)|J_2|d\xi\right]q_k^{ae} \quad i,j = 1,2 \tag{5.9}$$

Here ξ is the same variable it was in traditional BEM; local to the element and ranging from -1 to 1. $|J|$ is also unchanged and defined as in (4.8). $|J_2|$ is the Jacobian of the change from η to ξ, since R had been defined respecting η and is used in relation to ξ in this equation. As in 4.7, d_k^{ae} is the displacement coefficient at control point a of the element e in direction k. In the summations, not all the n elements must be computed, since only $p+1$ will be different from zero.

This situation is very similar to the one reached in traditional BEM but now the variables are **d** and **q**. After setting the relationship n times in different points, the system of equations will be built:

$$\mathbf{H'd} = \mathbf{G'q}, \tag{5.10}$$

and, after some extra considerations, solved.

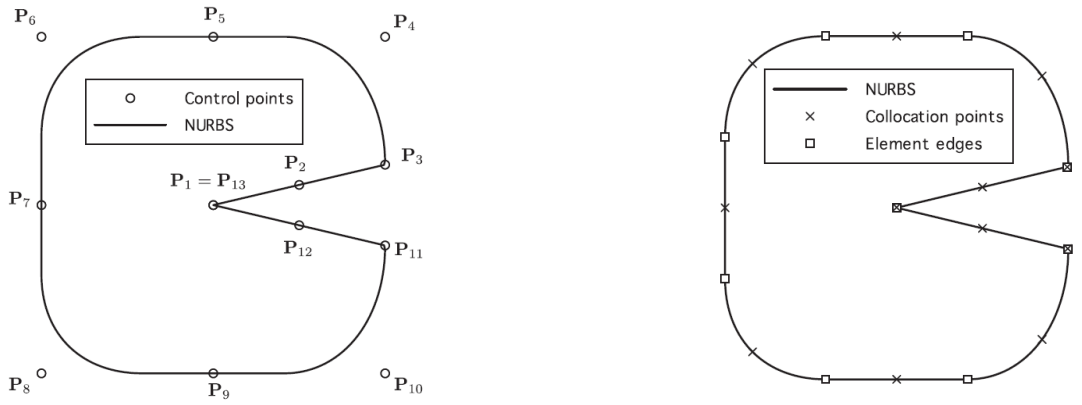
5.2.1 Collocation points

The first difficulty to face is the choice of the collocation points, that is, where to apply the forces for the fundamental solutions to work. While in traditional BEM the element nodes were the perfect candidates, here their akin control points are not so fitting since they will not generally be in the boundary.

Although the control points could be used even though the drawbacks, in this work the Greville abcissae points have been chosen [9], whose coordinates in parametric space are:

$$\eta_a^{'} = \frac{\eta_{a+1} + \eta_{a+2} + ... + \eta_{a+p}}{p} \qquad a = 1,2,...N. \tag{5.11}$$

These points are projections over the boundary of the control ones, as depicted in Figure 5.3:



(a) Control points. **(b)** Collocation points.

Figure 5.3 Greville abcissae collocation points. Figure taken from [19].

Again, this is not the only possible option, but it has been chosen following Simpson's work ([19]).

5.2.2 Boundary conditions

The current systems of equations has **d** and **q** as unknowns variables. Unlike the previous chapter, where the matrices could be reordered attending to what was given by the boundary conditions, in this case both vectors are completely unknown, for the gathered data is related to **u** and **p**. However, (5.7) and (5.8) allow the variables to be transformed. Two matrices **A** and **B** can be built, whose terms will be the different NURBS evaluated at the various collocation points, so that:

$$\mathbf{u} = \mathbf{Cd} \tag{5.12}$$

$$\mathbf{p} = \mathbf{Dq} \tag{5.13}$$

which, replaced into (5.10):

$$\mathbf{H'C^{-1}u} = \mathbf{G'D^{-1}p} \tag{5.14}$$

By joining terms and analogous to (4.9):

$$\mathbf{Hu} = \mathbf{Gp} \tag{5.15}$$

which can solved in the same way. Both **C** and **D** have inverse, since the relation between collocation and control points exists both ways and is unique.

5.2.3 Internal points

The computation of internal points was detailed in Equation 3.22. In the IGABEM the integration along the boundary will grant d and q terms, therefore the following adaptation must be made:

$$\sigma_{sl}^{y} + \int_{\Gamma} s_{slk}^{*} C_{sub} d_k d\Gamma = \int_{\Gamma} d_{slk}^{*} D_{sub} q_k d\Gamma \tag{5.16}$$

where C_{sub} are submatrices of C, related to the element or knot span on which the integration is being carried. In this case there would be no difference between C_{sub} and D_{sub} their differences lies in sharp corners, a single element cannot contain any of them.

5.3 Numerical implementation

The numerical implementation of the IGABEM is, in general, very similar to the BEM one. Although the fact that the control and collocation points can lay anywhere makes it harder to handle them, the basic ideas remain the same. In this section, deeper attention will be focused on their differences rather than their similarities.

5.3.1 Rigid body considerations

As seen in the previous chapter, the matrix **H** allowed the computation of its diagonal to be carried out without using the c_{lk} terms. However, it is not that clear whether that reasoning can be done in equation (5.14), for **d** and **q** have no physical meaning.

Despite this, the rigid body displacement can be applied into (5.14). Since matrix **C** is made, by rows, of the NURB functions evaluated at the different collocation points, it must sum 1 by rows at stated by 5.3. It can be proven (see Apendix A) than this causes $\mathbf{C}^{-1}$ to sum 1 by rows too. Additionally, it can also be proven (see Apendix A) that this causes $\mathbf{H}'\mathbf{C}^{-1}$ to sum one by rows.

Looking at 5.14 and 5.15, this makes it mandatory for H to sum 1 by rows, and possible for the rigid body consideration to be applied.

5.3.2 Free term coefficient

Although the computation of the free term coefficient c_{lk} is not mandatory for the method, it has been included for comparing results. The computation of these terms is detailed in [16], [14], [15] and will be included in this work for completeness. As they state in their work, it is limited to materials with a plane of elastic symmetry coincident with the coordinate plane x3=0, in-plane and out-of plane elastic solutions being therefore uncoupled. Only in-plane solutions corresponding to plane strain or plane stress states are analysed here.

The equations to be used depend on the nature of the material.

If the solid is made of a *mathematically non-degenerate material* (μ_m are distinct):

$$c_{kl} = Re\left(\sum_{\alpha=1}^{2} \frac{1}{i\pi\kappa_\alpha} F_{l\alpha} E_{k\alpha} log\frac{z_\alpha^{(1)}}{z_\alpha^{(2)}}\right) \tag{5.17}$$

where:

$$z_\alpha^{(e)} = r_1^{(e)} + \mu_\alpha r_2^{(e)} \tag{5.18}$$

$$\mathbf{E} = \begin{bmatrix} \xi_1(\mu_1) & \xi_1(\mu_2) \\ \mu_1^{-1}\xi_2(\mu_1) & \mu_2^{-1}\xi_2(\mu_2) \end{bmatrix} \tag{5.19}$$

$$\mathbf{F} = \begin{bmatrix} -\mu_1 & -\mu_2 \\ 1 & 1 \end{bmatrix} \tag{5.20}$$

$$\kappa_\alpha = 2\sum_{k=1}^{2} E_{k\alpha} F_{k\alpha} \tag{5.21}$$

$$\xi_a(\mu_k) = \mu_k^2 S_{a1} - \mu_k S_{a6} + S_{a2} \tag{5.22}$$

and $r_i^{(e)}$ is the i component of the tangent to the boundary on the collocation point, either before the point (e=1) or after it (e=2). The terms $S_i j$ are the components of the **S** matrix as defined in 2.5.

On the contrary, if in case of a *mathematically degenerate material* (μ_m are repeated and referred simply as μ)

$$c_{kl} = Re\left(\sum_{\alpha,\beta=1}^{2} F_{l\alpha} E_{k\beta} G_{\alpha\beta}(r^{(e)})\right)\Bigg|_2^1 \tag{5.23}$$

where:

$$\mathbf{E} = \begin{bmatrix} \xi_1(\mu_1) & 2\eta_1(\mu,\bar{\mu}) - \xi_1(\mu) \\ \mu\xi_1(\mu_1) & \bar{\mu}(2\eta_1(\mu,\bar{\mu}) - \xi_1(\mu)) \end{bmatrix} \tag{5.24}$$

$$\eta_a(p,q) = pqS_{a1} - 0.5(p+q)S_{a6} + S_{a2} \tag{5.25}$$

$$\mathbf{F} = \begin{bmatrix} -\mu & -\bar{\mu} \\ 1 & 1 \end{bmatrix} \tag{5.26}$$

$$\mathbf{G(x)} = \frac{1}{i\pi\kappa} \begin{bmatrix} \bar{z}(\mathbf{x})z^{-1}(\mathbf{x}) & log(z(\mathbf{x})) \\ log(z(\mathbf{x})) & 0 \end{bmatrix} \tag{5.27}$$

$$\begin{aligned} z(\mathbf{x}) &= x_1 + \mu x_2 \\ \bar{z}(\mathbf{x}) &= x_1 + \bar{\mu} x_2 \end{aligned} \tag{5.28}$$

The other terms ($\mathbf{r}$, $\mathbf{S}$, ξ) are as defined in the non-degenerate case.

5.3.3 Singularities in matrix H

The reasoning behind 4.2.2 is no longer valid, since NURBS are not interpolatory at the nodes. In this work, the subtraction of singularity method (SST) has been employed ([8]). It is a method to carry out singular integrals in the principal Cauchy value sense, and its idea is based on weakening the singularity and splitting the integral into a singular and a regular one. Further details are given in the article. The original article covered the isotropic case for the BEM method, but it has been adapted for the IGABEM anisotropic case.

Alternatively, García-Sánchez proposes a different method for dealing with these singularities in [7] by a change of variable.

5.3.4 Refinement

As already mentioned, the NURBS may need an additional refinement to bring valid results. A perfect example is 6.1; in that problem NURBS can represent the geometry of the solid exactly with the use of only 4 elements. However, the displacement and tractions follow a more complex evolution, so a refinement must be made if better accuracy is desired.

In this work only h-refinement is treated. P-refinement is a valid option but, since the aim of the work is to compare quadratic examples of both methods, has been not contemplated.

This refinement is achieved by inserting new knots in the knot vector. This can be done without altering the curve at all.

If the original knot vector was $\Xi = (\eta_1,\eta_2,...\eta_{n+p+1})$, the new one, called extended knot vector, will be $\overline{\Xi} = (\overline{\eta}_1 = \eta_1,\overline{\eta}_2,...\overline{\eta}_{n+m+p+1} = \eta_{n+p+1})$. The new $m+n$ control points will be linear combinations from the original ones. The way to calculate this new control points and its due weights so the geometry remains unchanged can be consulted in [3]. An example of knot-insertion is depicted in Figure 5.4.

5.3.5 Code organization

Now a brief description of the programs will be made. A flux diagram will be adjunct as well, to help the understanding of their behaviour.

ELQUABEIGABEMANI

This is the main program, which calls the others in order as seen in Figure 5.5. As in the BEM code, it uses cell arrays to separate the data from one boundary to another.

Figure 5.5 has two significant differences with the flow diagram that was shown in the BEM chapter (Figure 4.6). Firstly, whenever a singular integral needs to be computed, two different subroutines (LOCINEQANI and SST) will be called. This is because the rigid body motion technique in IGABEM allows to skip the computation of the free terms but not of the singularities in the $\mathbf{H}'$ matrix. Secondly, there is an HREFINEMENT subroutine that refines the input boundary curve if needed. This makes the refinement of the mesh much easier than in the BEM case.

The rest of the diagram flow is pretty much analogous.

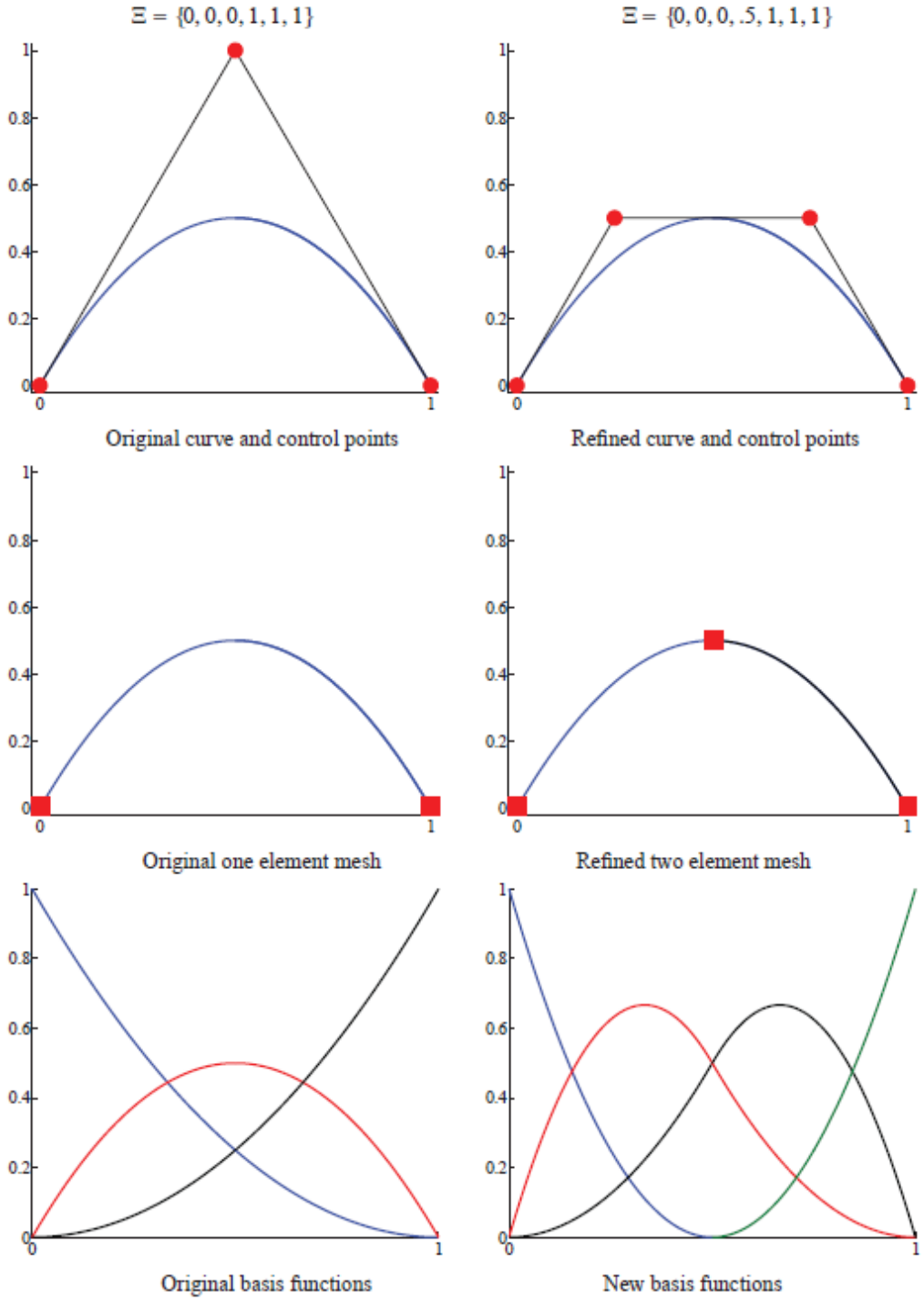


Figure 5.4 Knot insertion. Figure taken from [3].

INPUTEQIGABEMANI

This subprogram reads a .txt file with the input data. Additionally to what its BEM counterpart did, it also requires a refinement valor (which takes the zero value if no refinement is desired), the knot vector and the weights. Here the disparity between the number of displacement and tractions variable forces the requirement of separate vectors, KODE and KODE1. KODE is related to tractions and takes vale 0 if the traction of that position is unknown, and 1 otherwise. KODE1 is completely analogous, giving information about the displacement.

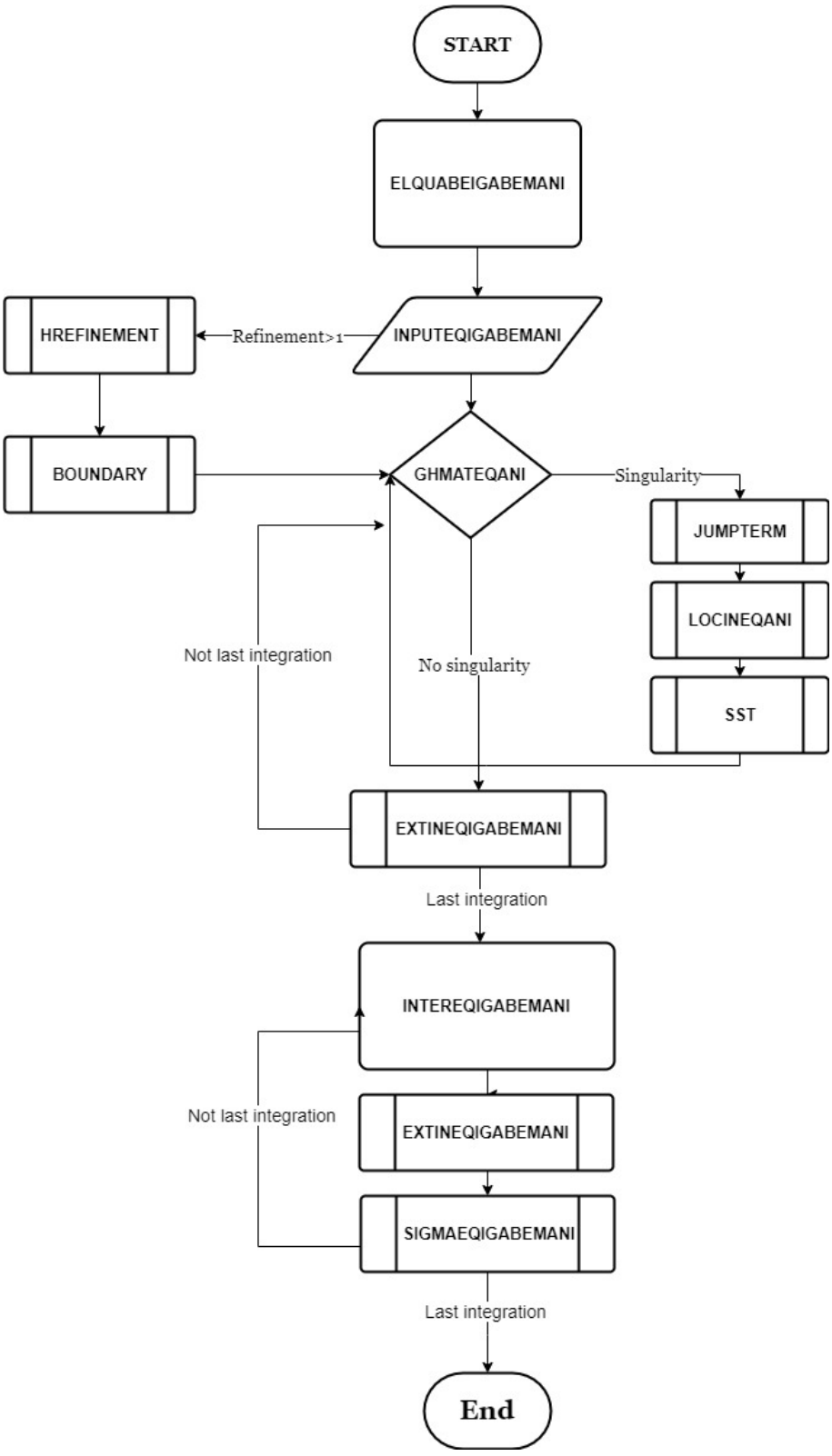


Figure 5.5 Flow diagram IGABEM.

HREFINEMENTANI

If the refinement value read in INPUTEQIGABEMANI was greater than one, this program inserts knots between the existing ones (refinement+1 knots will take the place of each previous knots). It also computes the new control points and weights to leave the geometry untouched, as commented in 5.3.4. This code is an

adaptation of the one provided by [18]

BOUNDARY

If the control points have been modified, the boundary conditions read by INPUTEQIGABEMANI is no longer enough, for now these new points require their data to be imposed. This subprogram creates the new boundary conditions and is only valid to the particular problem it was written for.

GHMATEQIGABEMANI

This program computes the different integrations needed for the matrices $\mathbf{H}'$ and $\mathbf{G}'$. For each of them, calls EXTINEQIGABEMANI if no singularities appear or both SST and LOCINEQIGABEM if they do. After this, computes the matrices $\mathbf{C}$ and $\mathbf{D}$, operates with them to reach equation (5.15) and interchanges $\mathbf{H}$ and $\mathbf{G}$ columns to create the final system.

The number of integrations that are carried out are one per collocation point per element, which is the number of times the loop in the flow diagram is repeated. After the last integration, the program can move on.

JUMPTERM

This program computes the c_{lk} terms to be included in the $\mathbf{H}'$ matrix.

EXTINEQIGABEMANI

This program computes the non-singular terms of both matrices using standard Gauss quadrature of ten points.

SST

This one computes the $\mathbf{H}'$ singular terms using the SST method. The method has been adjusted to tolerate freedom of singularity collocation, since the original document only treats BEM integration.

LOCINEQIGABEMANI

This program computes the $\mathbf{G}'$ singular terms in the very same way LOCINEQANI did, with a special logarithmic Gauss quadrature. The main difference here that now there are more than three possible node collocation. This has influence in the way the integral is split, but the procedure is similar and not worth repeating.

INTEREQIGABEMANI

Very similar to its BEM counterpart, this subprogram computes the results of the internal points with the help of EXTINEQIGABEMANI and SIGMAEQIGABEMANI.

The number of integrations that are carried out are one per internal point per element, which is the number of times the loop in the flow diagram is repeated. After the last integration, the program can move on.

SIGMAEQIGABEMANI

Also completely analogous to BEM SIGMAEQANI, this program computes the necessary terms $\mathbf{D}$ and $\mathbf{S}$ from equations in 3.3.2

6 Numerical Examples

In this chapter, practical examples will be solved by both methods (BEM and IGABEM) and the accuracy of their results tested.

6.1 Benchmark problem

The first problem solved is an orthotropic tube subjected to internal pressure. The material is glass-epoxy composite taken from [12] Because of the symmetry of the problem, only a quarter of the circle needs to be represented, as in Figure 6.1:

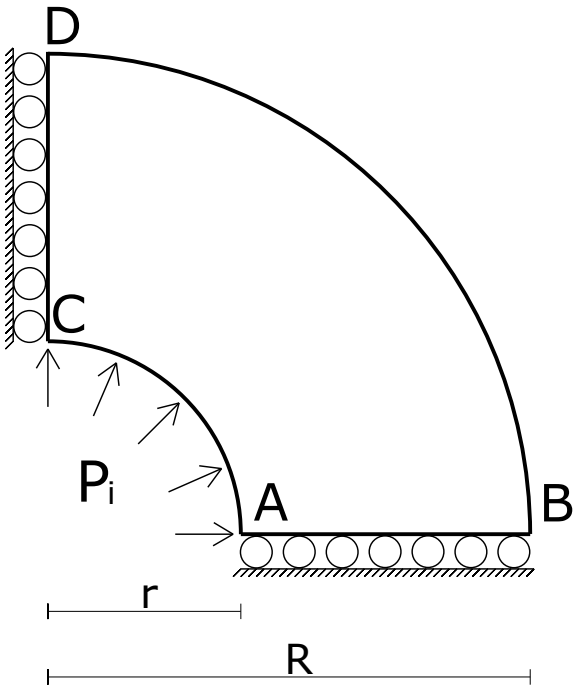


Figure 6.1 Orthotropic tube under internal pressure.

The data of the problem and the constants of the material, as per equation 2.5, are:

$$\begin{aligned} &r = 10\ mm \qquad R = 25\ mm \qquad P_i = 10\ MPa \\ &S_{11} = 7.2519 \cdot 10^{-6}\ MPa^{-1} \qquad S_{12} = -1.5229 \cdot 10^{-6}\ MPa^{-1} \\ &S_{22} = 6.9038 \cdot 10^{-5}\ MPa^{-1} \qquad S_{33} = 1.7114 \cdot 10^{-4}\ MPa^{-1} \end{aligned} \tag{6.1}$$

The results of both methods will be compared with those computed by ANSYS APDL using "Quad 4 node 182" as element type and 2771 total number of elements (see Figure. 6.2). The results in ANSYS will be

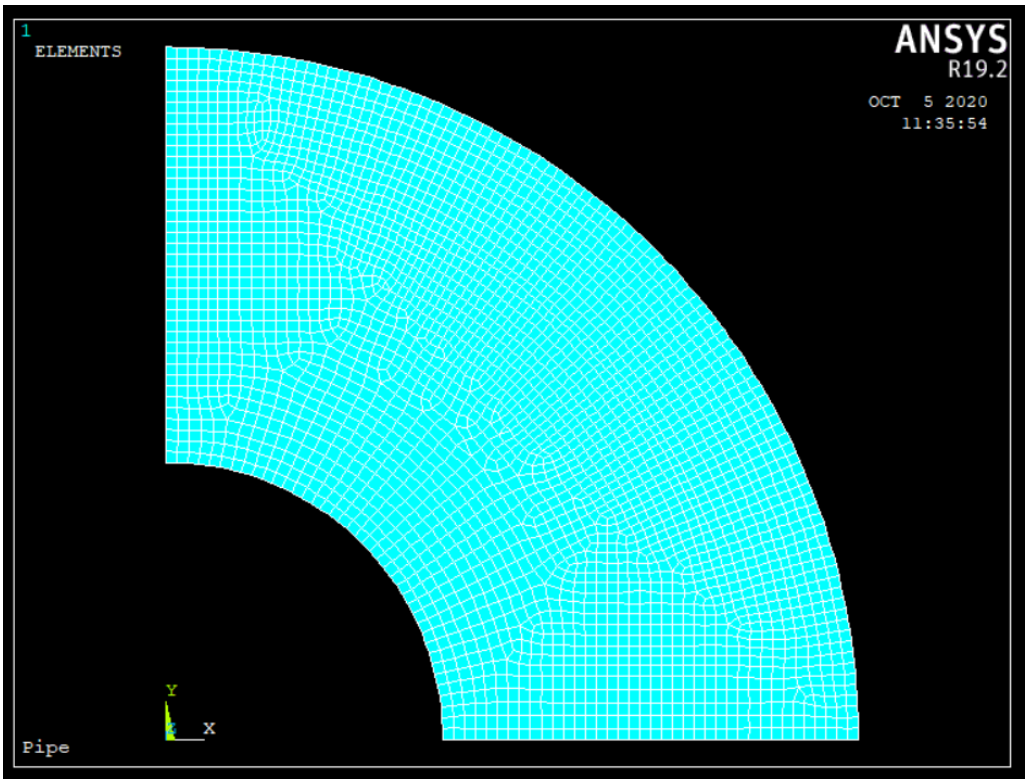
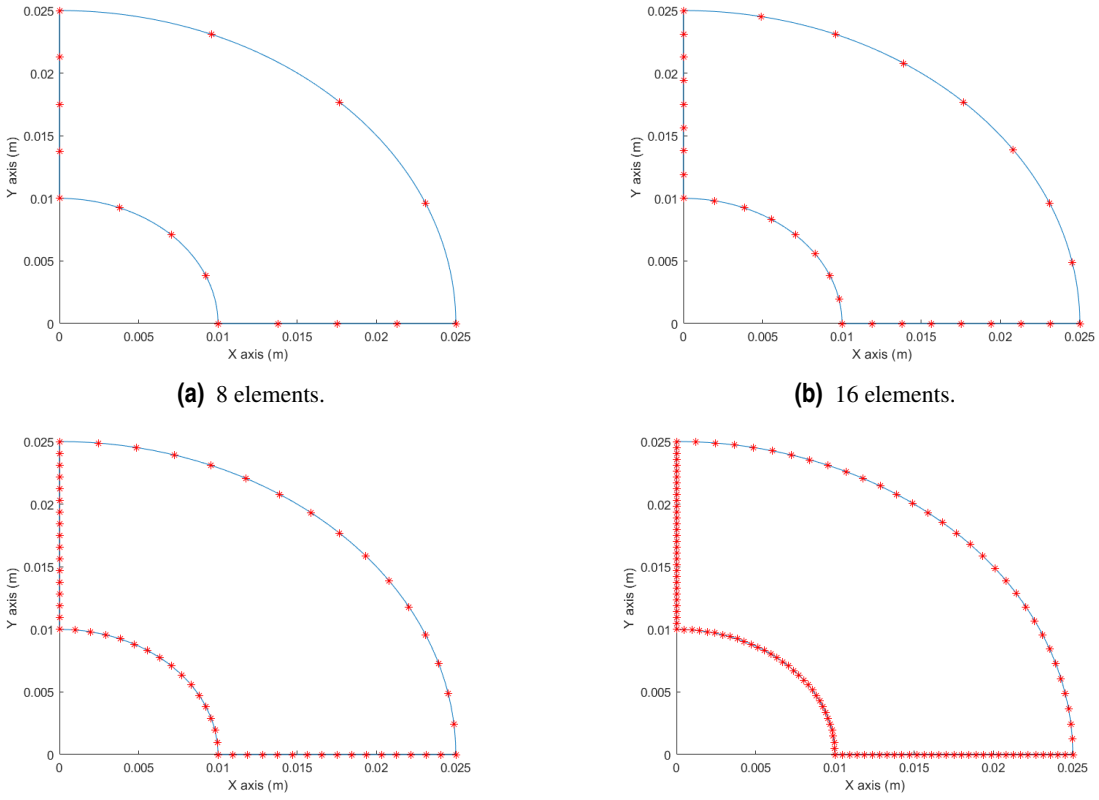


Figure 6.2 Model in ANSYS for comparison of results.



(a) 8 elements.

(b) 16 elements.

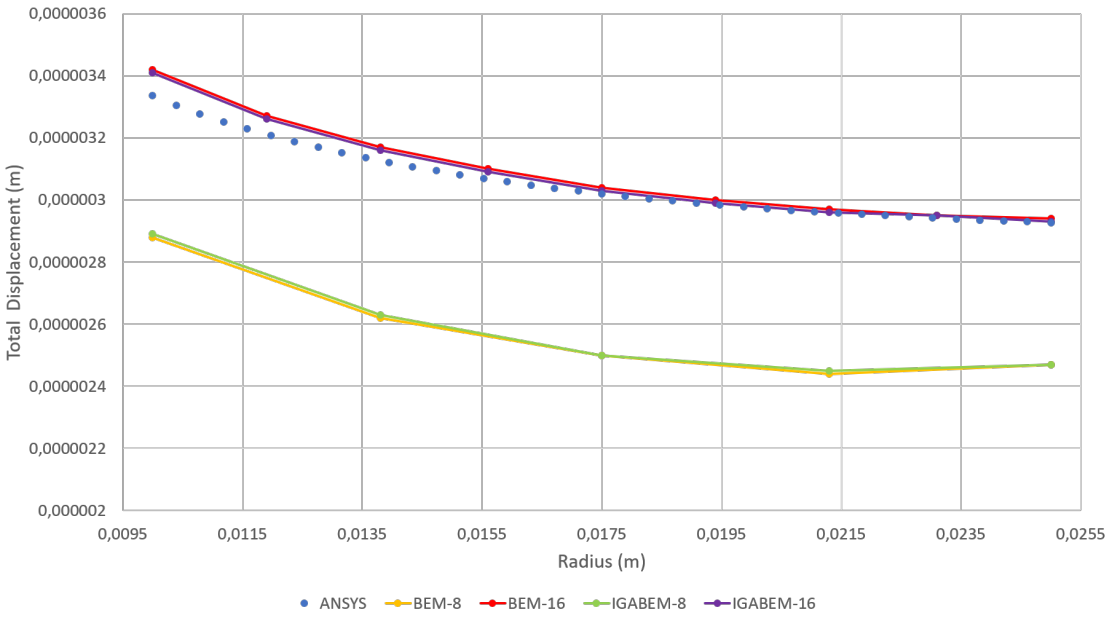
(c) 32 elements.

(d) 64 elements.

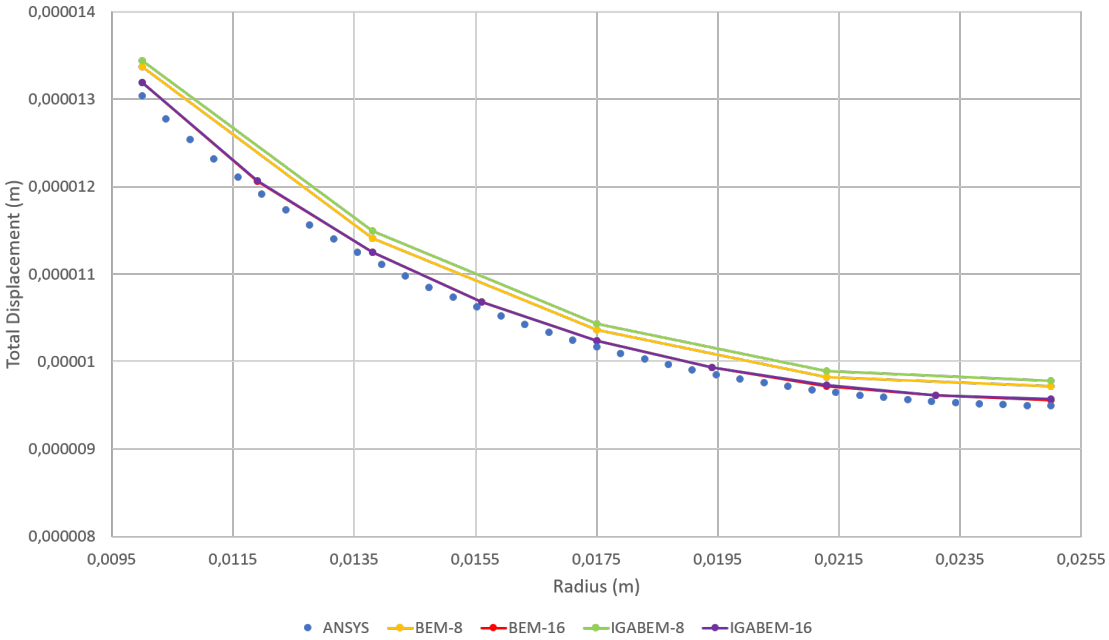
Figure 6.3 Pipe mesh for different refinements. Collocation points as red dots and geometry as blue lines.

compared with the results of MATLAB, both working under the plane strain assumption (meshes shown in Figure 6.3).

Firstly the displacements and stresses of the two sides will be compared with the MATLAB program for a low number of degrees of freedom, 2 and 4 elements per side (8 and 16 total elements in the boundary, respectively). Figure 6.4 shows the displacement values, and Figure 6.5 depicts the main principal stress. Tables 6.1, 6.2 and 6.3 show the difference in absolute value between MATLAB and ANSYS up to the first three significant digits.



(a) Displacements along the A-B side..



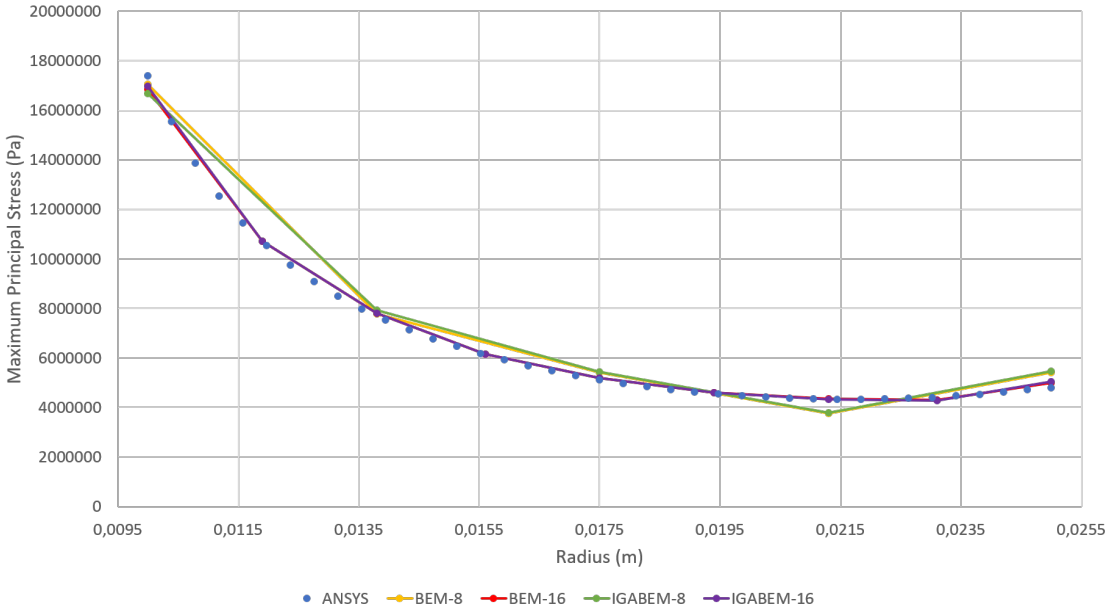
(b) Displacements along the C-D side.

Figure 6.4 Displacements on the boundary.

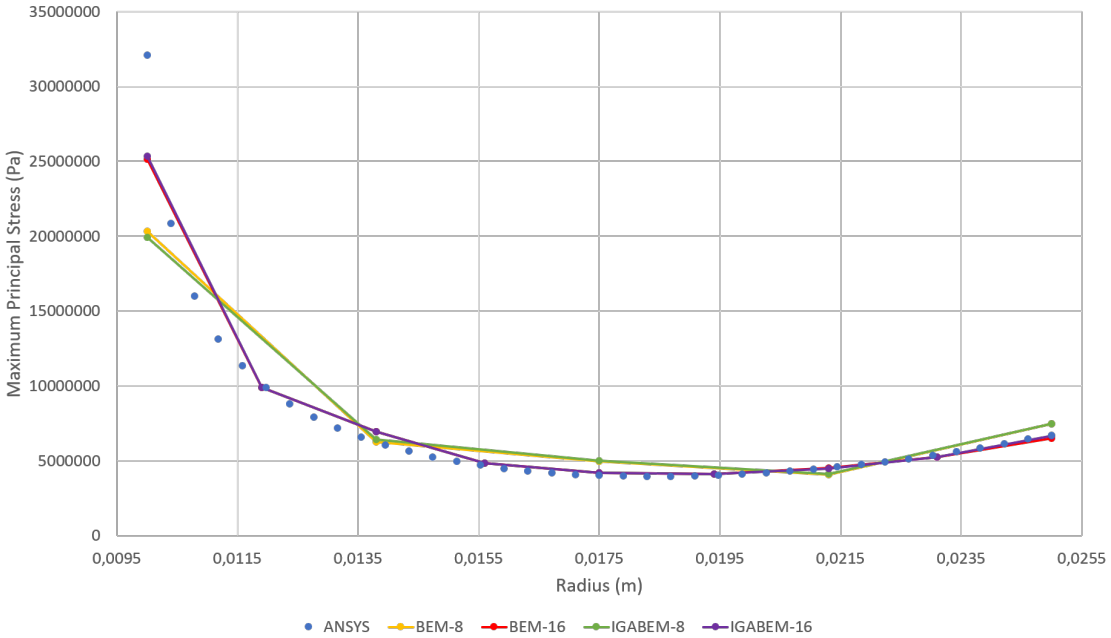
It is interesting to note that stress at points A and C are the least accurate in both methods, which is problematic because they present the highest stresses and displacements and are thus arguably the most important points on the solid. Figure 6.6 shows the evolution of the MATLAB results as the mesh becomes

Point	ANSYS (10^{-6}m)	BEM 8 (10^{-6}m)	Diff (%)	IGABEM 8 (10^{-6}m)	Diff (%)	BEM 16 (10^{-6}m)	Diff (%)	IGABEM 16 (10^{-6}m)	Diff (%)
A	3.33	2.9	13.5	2.89	13.2	3.42	2.7	3.41	2.4
B	2.93	2.5	15.1	2.47	15.7	3.04	3.8	3.03	3.4
C	13.0	13.4	2.8	13.4	3.1	13.2	1.2	13.2	1.2
D	9.49	9.7	3.0	9.78	3.1	9.56	0.7	9.57	0.8

Table 6.1 Comparison of the boundary displacement.



(a) Main principal stress along the A-B side.



(b) Main principal stress along the C-D side.

Figure 6.5 Stress on the boundary.

more refined. It shows the nodes in the case of BEM and the collocation points in the IGABEM case (they have been made to be coincident). It is clear that all results tend increasingly closer to the ANSYS result.

Point	ANSYS (MPa)	BEM 8 (MPa)	Diff (%)	IGABEM 8 (MPa)	Diff (%)	BEM 16 (MPa)	Diff (%)	IGABEM 16 (MPa)	Diff (%)
A	17.4	17.0	2.30	16.7	4.02	17.0	2.30	16.7	4.02
B	4.79	5.42	13.2	5.46	13.1	4.98	3.97	5.03	5.01
C	32.1	20.3	36.8	19.9	38.0	25.1	21.8	25.4	20.9
D	6.70	7.45	11.2	7.47	11.5	6.51	2.84	6.65	0.75

Table 6.2 Comparison of the boundary stress part 1.

Point	ANSYS (MPa)	BEM 32 (MPa)	Diff (%)	IGABEM 32 (MPa)	Diff (%)	BEM 64 (MPa)	Diff (%)	IGABEM 64 (MPa)	Diff (%)
A	17.4	17.2	1.15	17.2	1.15	17.2	1.15	17.2	1.15
B	4.79	4.80	0.21	4.8	0.21	4.78	0.21	4.78	0.21
C	32.1	28.4	11.5	28.4	11.5	30.4	5.30	30.4	5.30
D	6.70	6.52	2.69	6.52	2.69	6.49	3.13	6.49	5.30

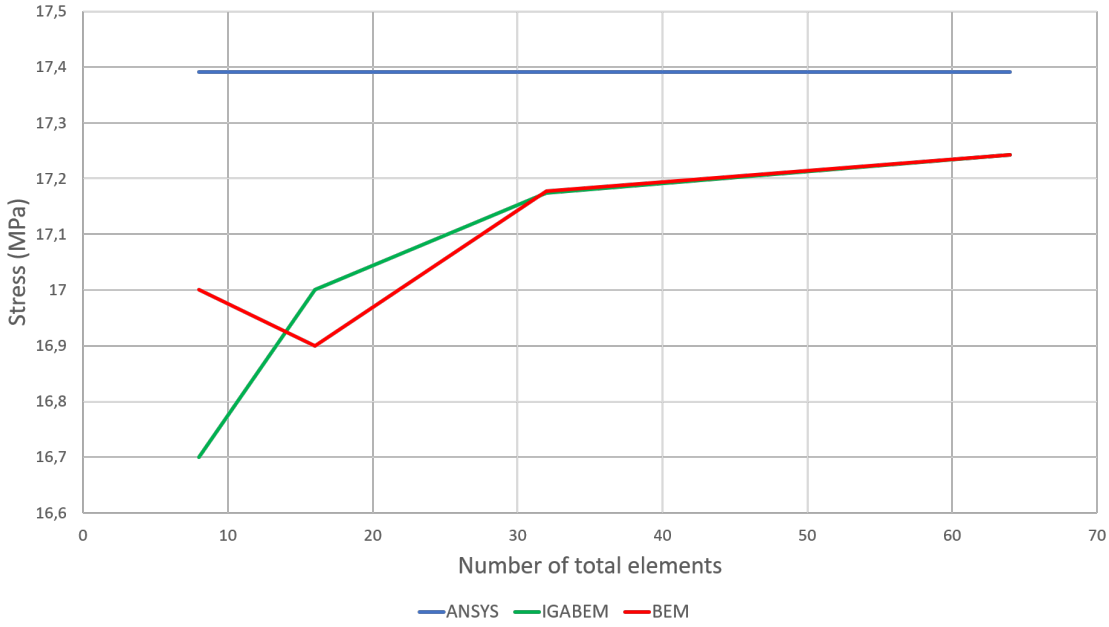
Table 6.3 Comparison of the boundary stress part 2.

In regards to the computation time, Table 6.4 shows comparative times for solving the boundary of the meshes in Figure 6.3:

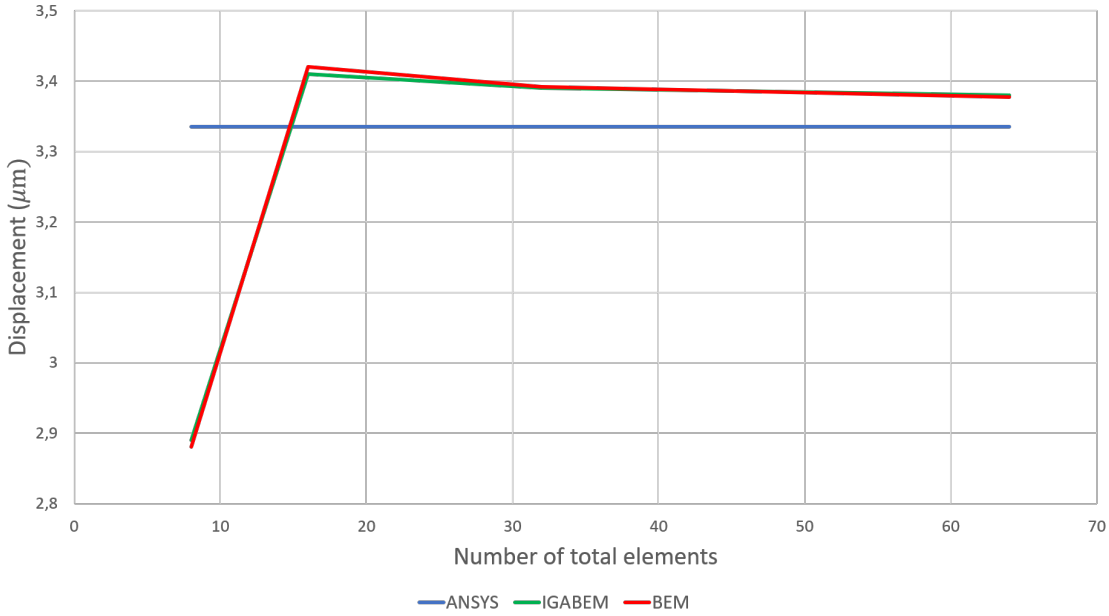
Method	8 elements (s)	16 elements (s)	32 elements (s)	64 elements (s)
BEM	1.60	3.07	8.15	27.43
IGABEM	4.91	11.61	34.65	103.49

Table 6.4 Comparison of the computation time.

It is clear that the IGABEM is computationally more costly than the BEM. A high percentage of the computation time is spent on the calculation of the NURB functions (Equation 5.2). As an example, it takes around 87 % of the total time in the case of the 32-element mesh.



(a) Stress at A.

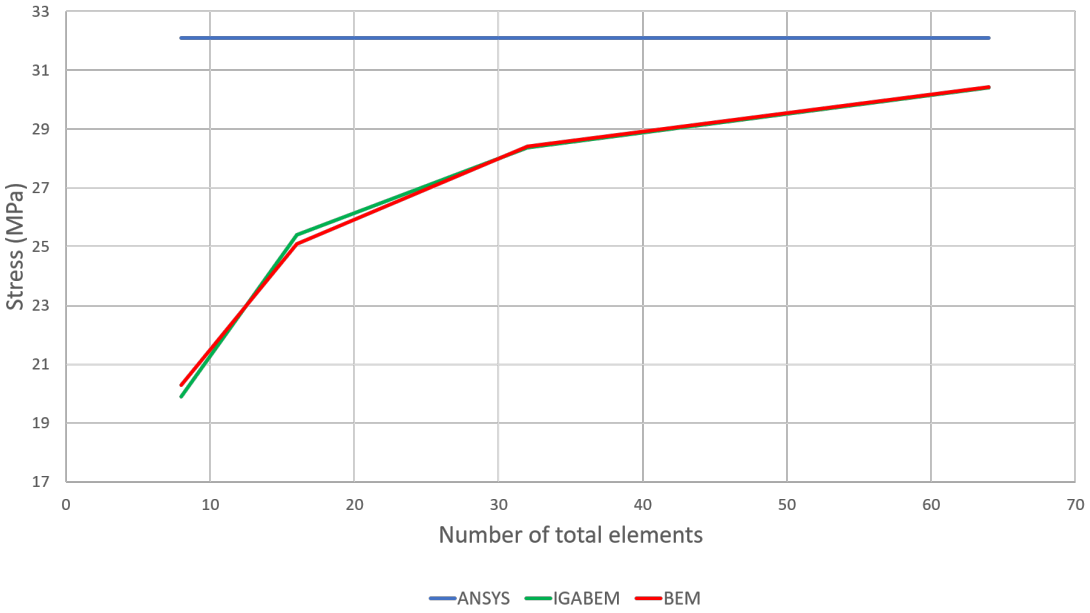


(b) Displacement at A.

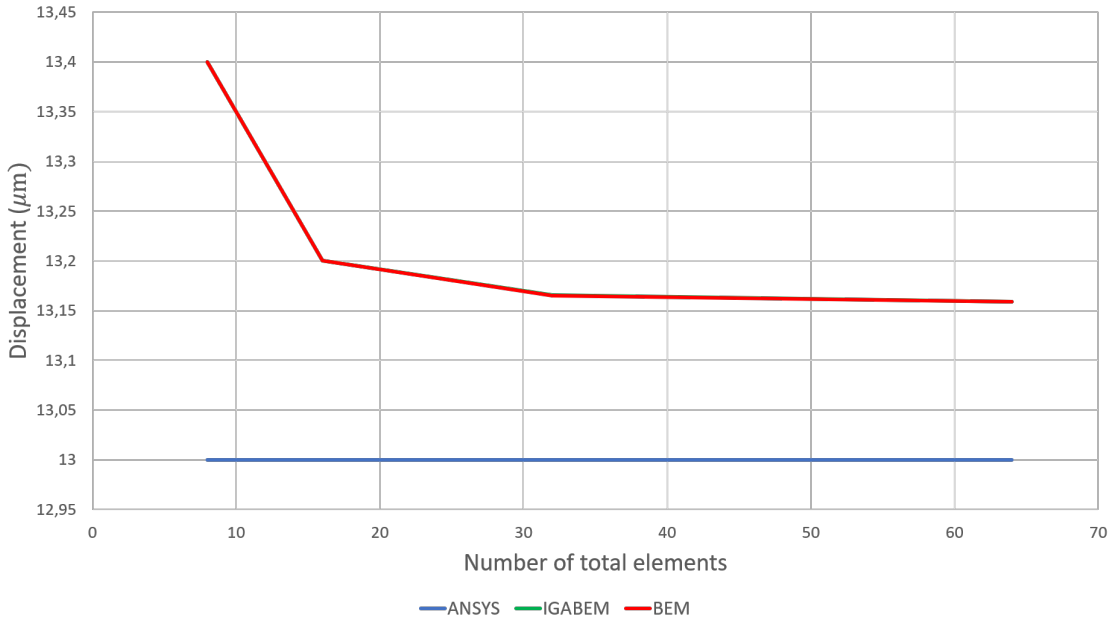
6.2 Plate with hole

The second problem to be solved is a thin plate with a hole in it. The geometry and loads, symmetry conditions included (minus the movement restriction in the direction perpendicular to the paper, which has not been included in the drawn), are represented in Figure 6.7 where:

$$\begin{array}{lll} r = 25\ mm & h = 200\ mm & c = 100\ mm \\ w = 50\ mm & P = 1\ MPa & \end{array} \tag{6.2}$$



(c) Stress at C.



(d) Displacement at C.

Figure 6.6 Evolution of the results as the mesh is refined..

The material is the same as in the previous example, whose stiffness matrix is (filling out the components that were not specified before), if fully shown (in MPa):

$$C = \begin{bmatrix} 138540 & 3060 & 0 & 0 & 0 & 0 \\ & 14550 & 0 & 0 & 0 & 0 \\ & & 14550 & 0 & 0 & 0 \\ & & & 5840 & 0 & 0 \\ & \text{sym.} & & & 5840 & 0 \\ & & & & & 5840 \end{bmatrix} \tag{6.3}$$

In order to use this material as the most generic anisotropic, it has been rotated 45° counter-clockwise in

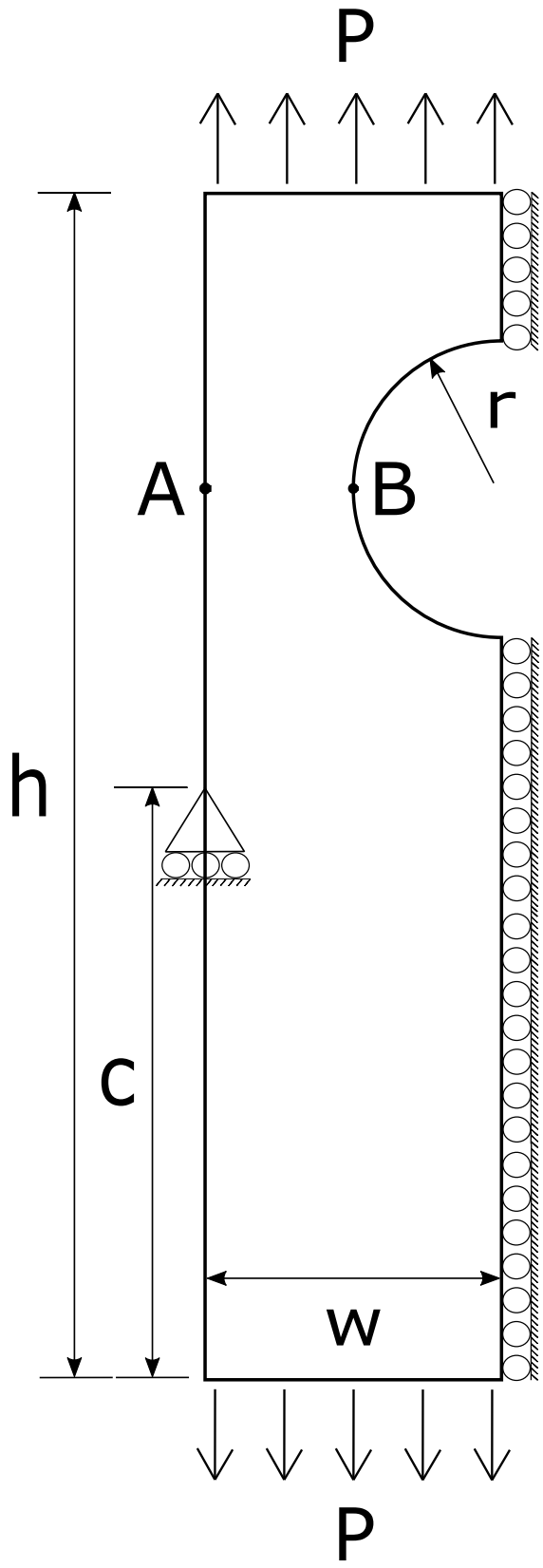


Figure 6.7 Thin plate with hole.

regards to each of the three x,y,z axis. The new behaviour matrix, whose transformation is detailed in [1], is

(in MPa):

$$C' = \begin{bmatrix} 21166 & 8592 & 16620 & -11037 & -10404 & 7580 \\ & 22248 & 17702 & -10787 & -12184 & 8121 \\ & & 44519 & -21790 & -22555 & 15678 \\ & & & 21312 & 15296 & -10878 \\ & \text{sym.} & & & 22394 & -11261 \\ & & & & & 13367 \end{bmatrix} \tag{6.4}$$

The results in MATLABS will again will be compared with those of ANSYS APDL ("Quad 4 node 182" as element type and 8805 total number of elements), both programs working under the assumption of plane stress. The meshes can be seen in Figure 6.8 and Figure 6.9.

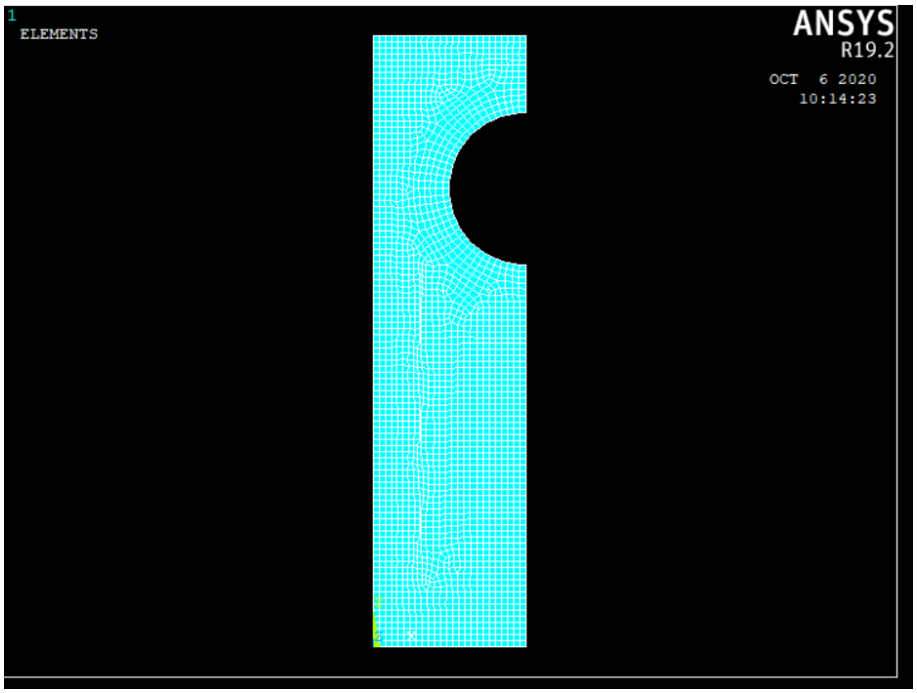


Figure 6.8 ANSYS mesh.

The main principal stress for the line between A and B is shown in Figure 6.10 and Table 6.5.

X axis (m)	ANSYS (MPa)	BEM (MPa)	Diff (%)	IGABEM (MPa)	Diff (%)
0.02	0.45841	0.5396	17.71	0.5162	12.61
0.12	1,7402	1.7587	1.06	1.7059	1.97
0.23	4.1261	4.165	0.94	4.2446	2.87

Table 6.5 1st Principal Stress along the AB line.

The total displacement in the plane for the line between A and B is shown in Figure 6.11 and Table 6.6.

X axis (m)	ANSYS (10E-4 m)	BEM (10E-4 m)	Diff (%)	IGABEM (10E-4 m)	Diff (%)
0.02	0.7307	0.7346	0.5364	0.7227	1.0876
0.12	0.7391	0.7408	0.2299	0.7287	1.4020
0.23	0.7678	0.7422	3.3302	0.7356	4.1847

Table 6.6 Total in-plane displacement along the AB line.

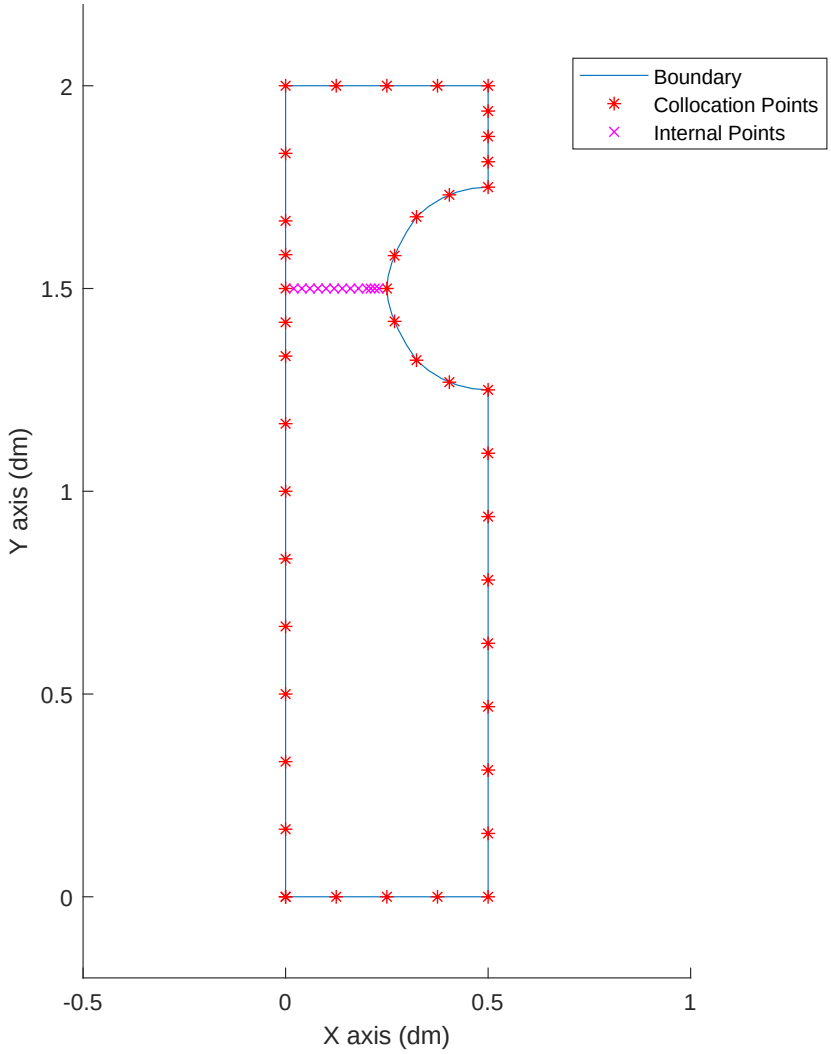


Figure 6.9 MATLAB mesh.

It is interesting to note that, since the program for computing internal points is not prepared to deal with singular integrals, the results become inaccurate when they are too close to the boundary. The distance from which they become unreliable is related to the distance between nodes in the close elements, so a finer mesh allows for smaller precision near the boundary. Figure 6.11 and Table 6.6 show that the accuracy of the solution in displacements decays as the points approach the boundary, although the error doesn't go over 5 %. A refinement in MATLAB has been proven to improve these differences (and the shape of the curves become more like the ANSYS one) although the results being shown have been selected to depict than even a not very dense mesh gives very accurate results away from the boundary and reasonably accurate results close to the boundary.

In any way, this example proves that the results of the program are quite good for problems with the most generic anisotropic materials, and with it both plane stress and plane strain have been covered.

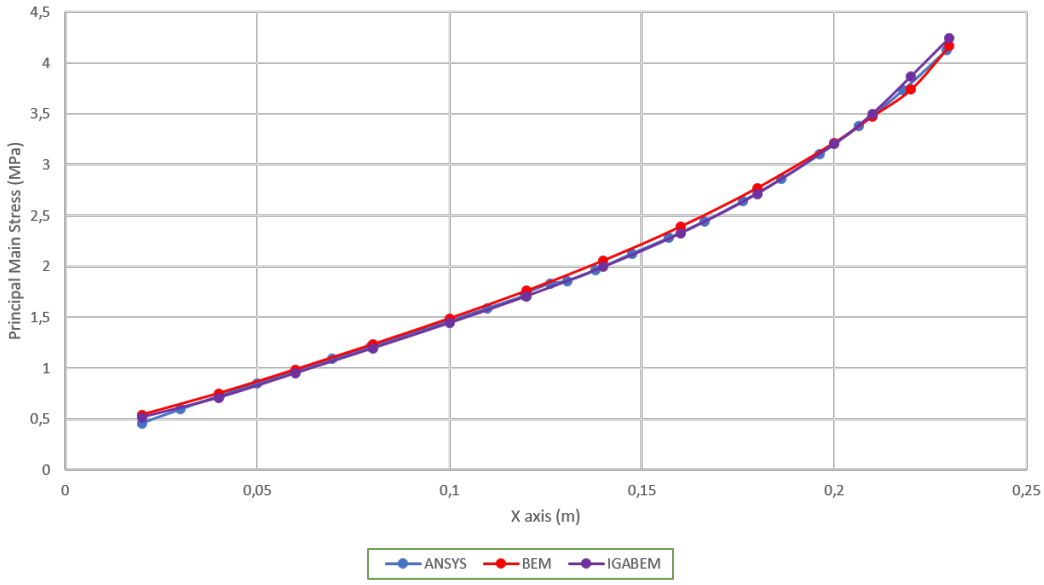


Figure 6.10 1st Principal Stress along the AB line.

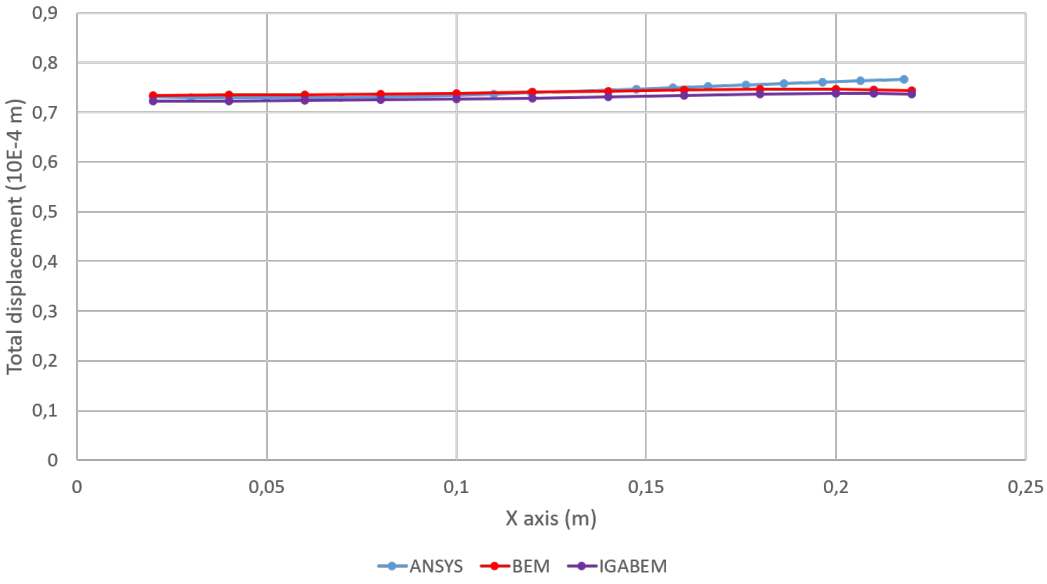


Figure 6.11 Total in-plane displacement along the AB line.

7 Conclusion and future work

In this work, the IGABEM method has been successfully implemented into a MATLAB code to solve practical problems of elasticity.

The theory behind the BEM equations have been reasoned, using the elastic equations. The working of the BEM has also been explained with detail, having focused additional attention into the problematic parts of it, such as the singular integrations. Later, these steps have been repeated with the IGABEM. The differences with the BEM have initially presented a difficulty, such as the assignment of collocation points or the application of boundary conditions. These problems have been studied and solved, allowing the IGABEM code to work.

There are several interesting results in the IGABEM. The rigid body considerations has been mathematically proved and tested in the code, in combination to the adaptation of the SST method.

The features of both methods have been tested. There has not been found a great difference in the accuracy of the solutions, for both methods brought relatively exact results without the need of a large number of elements. The main disparity has been found to lay in the computation and preparation time. While the IGABEM takes longer to compute, its refinement is significantly simpler and easier to do in this program. The second greatest advantage of the IGABEM over the BEM, the precision of the geometry, has not proved to have an decisive impact in the solution of these particular solutions. This is not very surprising since both geometries are quite simple and can be accurately represented with the shape forms in the BEM.

In regards to future work, there are several subjects that hold great potential.

Firstly, the code could be adapted to work with piezoelectric materials, a subset of anisotropic materials that create an electric voltage when subjected to stress and, inversely, suffer stress when subjected to voltage.

Secondly, it could be made to solve 3D problems, so its applicability would be extended.

Additionally, there is the possibility to improve the computation of internal points so that they can cope with singular integrals when the points are near the boundary, to avoid loses of accuracy as seen in Section 6.2. Another addition that could improve on the accuracy of this points is the implementation of other splines, such as discussed in the motivation section. This could allow for a local refinement near the internal point, and thus avoid the singularity.

Appendix A
Demonstrations

A.0.1 Property

Let **C=BA** be square matrices of order n.

$$\sum_{j=1}^{n} c_{ij} = \sum_{j=1}^{n} b_{ij} \left(\sum_{k=1}^{n} a_{jk} \right) \tag{A.1}$$

Demonstration:

$$\begin{aligned} c_{i1} &= b_{i1}a_{11} + ... + b_{in}a_{n1} \\ &\vdots \\ c_{in} &= b_{i1}a_{1n} + ... + b_{in}a_{nn} \end{aligned} \tag{A.2}$$

Summing all terms in A.2 $c_{i1} + ... + c_{in} = b_{i1}(a_{11} + ... + a_{1n}) + ... + b_{in}(a_{n1} + ... + a_{nn})$ which proves the property.

A.0.2 Conclusion 1

If **A** and **B** sum 1 by rows, **C** also does.

Demonstration:

$$\sum_{j=1}^{n} c_{ij} = \sum_{j=1}^{n} b_{ij} \left(\sum_{k=1}^{n} a_{jk} \right) = \sum_{j=1}^{n} b_{ij} = 1 \tag{A.3}$$

A.0.3 Conclusion 2

If **A** sums 1 by rows and has an inverse, $\mathbf{A^{-1}}$ also does:

Demonstration:

$$\begin{aligned} &\mathbf{I} = \mathbf{A^{-}1A} \\ &1 = \sum_{j=1}^{n} b_{ij} (\sum_{k=1}^{n} a_{jk}) = \sum_{j=1}^{n} b_{ij} = 1 \end{aligned} \tag{A.4}$$

Appendix B
Codes

B.1 ELQUABEIGABEMANI

```
%ELQUABEIGABEM
clc
clear all
close all

% This problem solves two dimensional elastic problems using quadratic
% boundary elements.
tic
[X,Y,displacements,tractions,CX,CY,N,p,knotVec,weight,L,refinement,NN,Amat,Qmat
    ,Lmat,mu_k,Ajump,Bjump,K,D,Cmat]=INPUTEQIGABEMANI;
if refinement>1
   [X,Y,knotVec,weight,N]=HREFINEMENT(knotVec,weight,refinement,X,Y,NN);
end
[C,FI,elRanges,collocPts,active,Afinal,Bfinal,KODEu,DFIu,KODE1v,DFI1v,Gs,
    collocPtsX,collocPtsY]=GHMATEQIGABEMANI(X,Y,displacements,tractions,N,p,
    knotVec,weight,refinement,NN,Amat,Qmat,Lmat,mu_k,Ajump,Bjump,K,D);
SOL=C\FI;
toc
[SSOL,DSOL,U,T]=INTEREQIGABEMANI(N,X,Y,SOL,L,CX,CY,elRanges,knotVec,weight,NN,
    active,Afinal,Bfinal,KODEu,DFIu,KODE1v,DFI1v,Gs,Amat,Qmat,Lmat,mu_k,Cmat,
    collocPtsX,collocPtsY);
save('IGABEM.mat')
```

B.2 INPUTEQIGABEMANI

```
function [X,Y,displacements,tractions,CX,CY,N,p,knotVec,weights,L,refinement,NN
    ,Amat,Qmat,Lmat,mu_k,Ajump,Bjump,K,D,Cmat]=INPUTEQIGABEMANI


%NE=NUmber of boundary elements.
%N=Number of boundary nodes (=2*NE).
%L=Number of internal points.
%GE=Shear modulus.
%XNU=Poisson ratio.

clear all
```

```
archivo=fopen('Anisotropic Mahajerin IGABEM plus girado.txt','r');

%Read number of boundary elements and internal points.
NN=fscanf(archivo,'%g',[1,1]);
for n=1:NN
NE(n)=fscanf(archivo,'%d',[1,1]);
N(n)=NE(n)*2;
end
L=fscanf(archivo,'%d',[1,1]);
% GE=fscanf(archivo,'%g',[1,1]);
% XNU=fscanf(archivo,'%g',[1,1]);
%Read level of refinement.
refinement=fscanf(archivo,'%d',[1,1]);
Cmat=zeros(6,6);
Cmat(1,1)=fscanf(archivo,'%g',[1,1]);
Cmat(1,2)=fscanf(archivo,'%g',[1,1]);
Cmat(1,3)=fscanf(archivo,'%g',[1,1]);
Cmat(1,4)=fscanf(archivo,'%g',[1,1]);
Cmat(1,5)=fscanf(archivo,'%g',[1,1]);
Cmat(1,6)=fscanf(archivo,'%g',[1,1]);
Cmat(2,1)=Cmat(1,2);
Cmat(2,2)=fscanf(archivo,'%g',[1,1]);
Cmat(2,3)=fscanf(archivo,'%g',[1,1]);
Cmat(2,4)=fscanf(archivo,'%g',[1,1]);
Cmat(2,5)=fscanf(archivo,'%g',[1,1]);
Cmat(2,6)=fscanf(archivo,'%g',[1,1]);
Cmat(3,1)=Cmat(1,3);
Cmat(3,2)=Cmat(2,3);
Cmat(3,3)=fscanf(archivo,'%g',[1,1]);
Cmat(3,4)=fscanf(archivo,'%g',[1,1]);
Cmat(3,5)=fscanf(archivo,'%g',[1,1]);
Cmat(3,6)=fscanf(archivo,'%g',[1,1]);
Cmat(4,1)=Cmat(1,4);
Cmat(4,2)=Cmat(2,4);
Cmat(4,3)=Cmat(3,4);
Cmat(4,4)=fscanf(archivo,'%g',[1,1]);
Cmat(4,5)=fscanf(archivo,'%g',[1,1]);
Cmat(4,6)=fscanf(archivo,'%g',[1,1]);
Cmat(5,1)=Cmat(1,5);
Cmat(5,2)=Cmat(2,5);
Cmat(5,3)=Cmat(3,5);
Cmat(5,4)=Cmat(4,5);
Cmat(5,5)=fscanf(archivo,'%g',[1,1]);
Cmat(5,6)=fscanf(archivo,'%g',[1,1]);
Cmat(6,1)=Cmat(1,6);
Cmat(6,2)=Cmat(2,6);
Cmat(6,3)=Cmat(3,6);
Cmat(6,4)=Cmat(4,6);
Cmat(6,5)=Cmat(5,6);
Cmat(6,6)=fscanf(archivo,'%g',[1,1]);

% Change of constants if plane stress

Cmat2=Cmat;
for i=1:6
    for j=1:6
        Cmat(i,j)=Cmat2(i,j)-Cmat2(i,3)*Cmat2(3,j)/Cmat2(3,3);
```

```
    end
end

%Read curve degree.
p=fscanf(archivo,'%d',[1,1]);
%Read control points coordinates.

for n=1:NN
coordenadas{n}=fscanf(archivo,'%g',[2*N(n),1]);
Coor{n}=[coordenadas{n}(1:2:2*N(n)),coordenadas{n}(2:2:2*N(n))];
end

%Read knotVec and weights.

for n=1:NN
knotVec{n}=fscanf(archivo,'%g',[N(n)+p+2,1]);
uniqueKnots{n}=unique(knotVec{n});
weights{n}=fscanf(archivo,'%g',[N(n)+1,1]);
end

%Read coordinates of the internal points.

coordenadas2=fscanf(archivo,'%g',[2*L,1]);
Coor2=[coordenadas2(1:2:2*L),coordenadas2(2:2:2*L)];

%Read boundary conditions

displacements=fscanf(archivo,'%g',6*sum(N));
tractions=fscanf(archivo,'%g')';

% Naming.

for n=1:NN
X{n}=Coor{n}(:,1);
Y{n}=Coor{n}(:,2);
X{n}(length(X{n})+1)=X{n}(1);
Y{n}(length(Y{n})+1)=Y{n}(1);
end

CX=Coor2(:,1);
CY=Coor2(:,2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% v=XNU;
% E=2*GE*(1+v);
% alfa=v/(1-2*v);
% const=E/(1+v);
% Cmat=zeros(6,6);
% Cmat(1,1)=const*(1+alfa);
% Cmat(1,2)=const*alfa; Cmat(2,1)=Cmat(1,2);
% Cmat(1,3)=const*alfa; Cmat(3,1)=Cmat(1,3);
% Cmat(2,2)=const*(1+alfa);
% Cmat(2,3)=const*alfa; Cmat(3,2)=Cmat(2,3);
% Cmat(3,3)=const*(1+alfa);
% Cmat(4,4)=const*0.5;
% Cmat(5,5)=const*0.5;
% Cmat(6,6)=const*0.5;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C2ij2=[Cmat(6,6),Cmat(6,2),Cmat(6,4);Cmat(2,6),Cmat(2,2),Cmat(2,4);Cmat(4,6),
    Cmat(4,2),Cmat(4,4)];
C1ij2=[Cmat(1,6),Cmat(1,2),Cmat(1,4);Cmat(6,6),Cmat(6,2),Cmat(6,4);Cmat(5,6),
    Cmat(5,2),Cmat(5,4)];
C2ij1=[Cmat(6,1),Cmat(6,6),Cmat(6,5);Cmat(2,1),Cmat(2,6),Cmat(2,5);Cmat(4,1),
    Cmat(4,6),Cmat(4,5)];
C1ij1=[Cmat(1,1),Cmat(1,6),Cmat(1,5);Cmat(6,1),Cmat(6,6),Cmat(6,5);Cmat(5,1),
    Cmat(5,6),Cmat(5,5)];
matrix=[-C2ij2\C2ij1,eye(3)/C2ij2;C1ij2/C2ij2*C2ij1-C1ij1,-C1ij2/C2ij2];
[AL,mu]=eig(matrix);

mu_k=diag(mu);

Lmat=AL(4:6,imag(mu_k)>0);
Amat=AL(1:3,imag(mu_k)>0);
mu_k=mu_k(imag(mu_k)>0);
Bmat=1i*Amat/Lmat;
Qmat=Amat\(eye(3)/(eye(3)/Bmat+eye(3)/conj(Bmat)));

subC=[Cmat(1,1),Cmat(1,2),Cmat(1,6);Cmat(1,2),Cmat(2,2),Cmat(2,6);Cmat(1,6),
    Cmat(2,6),Cmat(6,6)];
S=inv(subC);

Chi11=mu_k(1)^2*S(1,1)-mu_k(1)*S(1,3)+S(1,2);
Chi12=mu_k(2)^2*S(1,1)-mu_k(2)*S(1,3)+S(1,2);
Chi21=mu_k(1)^2*S(2,1)-mu_k(1)*S(2,3)+S(2,2);
Chi22=mu_k(2)^2*S(2,1)-mu_k(2)*S(2,3)+S(2,2);

if abs(mu_k(1)-mu_k(2))<1e-7 || abs(mu_k(1)-conj(mu_k(2)))<1e-7

    D=1;
    pjump=mu_k(1);
    Nu1=pjump*conj(pjump)*S(1,1)-0.5*(pjump+conj(pjump))*S(1,3)+S(1,2);
    Bjump=[-pjump,-conj(pjump);1,1];
    Ajump=[Chi11,2*Nu1-Chi11;pjump*Chi11,conj(pjump)*(2*Nu1-Chi11)];
    K=Ajump(1,1)*Bjump(1,2)+Bjump(1,1)*Ajump(1,2)+Ajump(2,1)*Bjump(2,2)+Bjump
        (2,1)*Ajump(2,2);

else

    D=0;
    Bjump=[-mu_k(1),-mu_k(2);1,1];
    Ajump=[Chi11,Chi12;Chi21/mu_k(1),Chi22/mu_k(2)];
    K(1)=2*(Ajump(1,1)*Bjump(1,1)+Ajump(2,1)*Bjump(2,1));
    K(2)=2*(Ajump(1,2)*Bjump(1,2)+Ajump(2,2)*Bjump(2,2));
end

end
```

B.3 GHMATEQIGABEMANI

```
function [C,FI,elRanges,collocPts,active,Afinal,Bfinal,KODEu,DFIu,KODE1v,DFI1v,
    Gs,collocPtsX,collocPtsY]=GHMATEQIGABEMANI(X,Y,displacements,tractions,N,p,
    knotVec,weight,refinement,NN,Amat,Qmat,Lmat,mu_k,Ajump,Bjump,K,D)

% This subroutine computes the G and H matrices and forms the system % C*X=FI
    ready to be solved.

% Computation of the geometry and plotting.

paso=0.2;
tam=0;
NT=sum(N);
totalactive=0;

for n=1:NN
    [~,~,aux]=unique(knotVec{n});
    knotunit{n}=aux-1;
    uniqueKnots{n}=unique(knotVec{n})';
    elRanges{n}=[uniqueKnots{n}(1:end-1)',uniqueKnots{n}(2:end)'];
    n1{n}=size(elRanges{n});
    tam=tam+n1{n}(1);
    connectivity{n}=zeros(N(n),size(uniqueKnots{n},2));
    for k=1:N(n)
         connectivity{n}(k,knotunit{n}(k)+1:knotunit{n}(k+3)+1)=1;
    end
    for k=1:size(uniqueKnots{n},2)-1
        active{n}(k)=find(connectivity{n}(:,k).*connectivity{n}(:,k+1),1);
        activeB{n}(k)=active{n}(k)+k-1;
    end
    totalactive=totalactive+size(elRanges{n},1);
    xi{n}=0:paso:knotVec{n}(end);
    R{n}=zeros(N(n)+1,numel(xi));
    dR{n}=zeros(N(n)+1,numel(xi));
    collocPts{n}=zeros(1,N(n)+1);
    index{n}=[1:N(n)];
for i=1:N(n)+1
collocPts{n}(i)=sum(knotVec{n}(i+1:i+p))/p;
end
for k=1:N(n)+1
    j=0;
    for aux=0:paso:knotVec{n}(end)
       j=j+1;
       [R{n}(k,j),dR{n}(k,j)]=NURBSbasis_matlab(k,p,aux,knotVec{n},weight{n});
    end
    figure(n)
   plot(xi{n},R{n}(k,:))
   hold on
    k
end
j=0;
x{n}=zeros(numel(xi{n}),1);
y{n}=zeros(numel(xi{n}),1);
for c=0:paso:knotVec{n}(end)
    j=j+1;
    pos=find(xi{n}>=c,1);
    for k=1:N(n)+1
    x{n}(j)=x{n}(j)+X{n}(k)*R{n}(k,pos);
```

```
    y{n}(j)=y{n}(j)+Y{n}(k)*R{n}(k,pos);
    end
end
figure(NN+1)
hold on
plot(x{n},y{n})
collocPtsX{n}=zeros(N(n)+1,1);
collocPtsY{n}=zeros(N(n)+1,1);
collocNURBS=zeros(N(n)+1,N(n)+1);
min=1;
for k=1:N(n)+1
     collocPtsX{n}(k)=0;
     collocPtsY{n}(k)=0;
    for j=1:N(n)+1
        if j>=min && j<=min+6
            collocNURBS(k,j)=NURBSbasis_matlab(j,p,collocPts{n}(k),knotVec{n},
                weight{n});
            if collocNURBS(k,j)==0 && collocNURBS(k,max(j-1,1))~=0
                min=max(j-4,1);
                break
            end
        end
        collocPtsX{n}(k)= collocPtsX{n}(k)+collocNURBS(k,j)*X{n}(j);
        collocPtsY{n}(k)= collocPtsY{n}(k)+collocNURBS(k,j)*Y{n}(j);
    end
end
hold on
plot(collocPtsX{n},collocPtsY{n},'*r')
end

% Compute the A and B matrices (and also AC);

for n=1:NN
    A{n}=zeros(3*N(n),3*N(n));
    AC{n}=zeros(3*N(n),3*N(n));
    B{n}=zeros(9*length(active{n}),9*length(active{n}));
    for i=1:3:3*N(n)
        for j=1:3:3*N(n)
            %A{n}(i,j)=NURBSbasis_matlab(ceil(j/2),2,collocPts{n}(ceil(i/2)),
                knotVec{n},weight{n});
        end
    end
    for i=2:3:3*N(n)
        for j=2:3:3*N(n)
            %A{n}(i,j)=NURBSbasis_matlab(ceil(j/2),2,collocPts{n}(ceil(i/2)),
                knotVec{n},weight{n});
        end
    end
    for i=1:3:3*N(n)
        for j=1:3:3*N(n)
            %AC{n}(i,j)=NURBSbasis_matlab(ceil(j/2),2,collocPts{n}(ceil(i/2)),
                knotVec{n},weight{n});
            AC{n}(i,j)=collocNURBS(ceil(i/3),ceil(j/3));
            AC{n}(i+1,j)=AC{n}(i,j);
            AC{n}(i+2,j)=AC{n}(i,j);
            AC{n}(i,j+1)=AC{n}(i,j);
            AC{n}(i+1,j+1)=AC{n}(i,j);
```

```
            AC{n}(i+2,j+1)=AC{n}(i,j);
            AC{n}(i,j+2)=AC{n}(i,j);
            AC{n}(i+1,j+2)=AC{n}(i,j);
            AC{n}(i+2,j+2)=AC{n}(i,j);
            A{n}(i,j)=AC{n}(i,j);
            A{n}(i+1,j+1)=AC{n}(i,j);
            A{n}(i+2,j+2)=AC{n}(i,j);
        end
    end
    A{n}(end-2,1)=NURBSbasis_matlab(N(n)+1,2,collocPts{n}(end-1),knotVec{n},
        weight{n});
    A{n}(end-1,2)=NURBSbasis_matlab(N(n)+1,2,collocPts{n}(end-1),knotVec{n},
        weight{n});
    A{n}(end,3)=NURBSbasis_matlab(N(n)+1,2,collocPts{n}(end-1),knotVec{n},
        weight{n});
    AC{n}(end-2,1:3)=NURBSbasis_matlab(N(n)+1,2,collocPts{n}(end-1),knotVec{n},
        weight{n});
    AC{n}(end-1,1:3)=NURBSbasis_matlab(N(n)+1,2,collocPts{n}(end-1),knotVec{n},
        weight{n});
    AC{n}(end,1:3)=NURBSbasis_matlab(N(n)+1,2,collocPts{n}(end-1),knotVec{n},
        weight{n});
    for i=1:3:9*length(active{n})
        for j=1:3:9*length(active{n})
            if ceil(j/3)==activeB{n}(ceil(i/9)) || ceil(j/3)==activeB{n}(ceil(i
                /9))+1 || ceil(j/3)==activeB{n}(ceil(i/9))+2
                B{n}(i,j)=NURBSbasis_matlab(active{n}(ceil(j/9))+floor(rem(j
                    -1,9)/3),2,collocPts{n}(active{n}(ceil(i/9))+floor(rem(i-1,9)
                    /3)),knotVec{n},weight{n});
            end
        end
    end
     for i=2:3:9*length(active{n})
        for j=2:3:9*length(active{n})
            if ceil(j/3)==activeB{n}(ceil(i/9)) || ceil(j/3)==activeB{n}(ceil(i
                /9))+1 || ceil(j/3)==activeB{n}(ceil(i/9))+2
                B{n}(i,j)=NURBSbasis_matlab(active{n}(ceil(j/9))+floor(rem(j
                    -1,9)/3),2,collocPts{n}(active{n}(ceil(i/9))+floor(rem(i-1,9)
                    /3)),knotVec{n},weight{n});
            end
        end
     end
    for i=3:3:9*length(active{n})
        for j=3:3:9*length(active{n})
            if ceil(j/3)==activeB{n}(ceil(i/9)) || ceil(j/3)==activeB{n}(ceil(i
                /9))+1 || ceil(j/3)==activeB{n}(ceil(i/9))+2
                B{n}(i,j)=NURBSbasis_matlab(active{n}(ceil(j/9))+floor(rem(j
                    -1,9)/3),2,collocPts{n}(active{n}(ceil(i/9))+floor(rem(i-1,9)
                    /3)),knotVec{n},weight{n});
            end
        end
    end
    if n==1
        Afinal=A{n};
        ACfinal=AC{n};
        Bfinal=B{n};
        Activefinal=active{n};
    else
```

```
        Afinal=blkdiag(Afinal,A{n});
        ACfinal=blkdiag(ACfinal,AC{n});
        Bfinal=blkdiag(Bfinal,B{n});
        Activefinal=[Activefinal,active{n}+Activefinal(end)+1];
    end
end

G=zeros(3*NT,9*totalactive);
H=zeros(3*NT,3*NT);

%
    ------------------------------------------------------------------------------------------------


% Compute the GW and HW matrices for each collocation point and each
% boundary element.
global NURBS dNURBS
NURBS=zeros(N(1)+1,(10*size(elRanges{NN},1)));
dNURBS=NURBS;
pGH=0;
for n=1:NN
     THETA{n}=zeros(N(n),1);
   for ll=1:N(n)
       pG=0;
       pH=0;

       % Calculate jump term

       if ll~=1
           if ismember(collocPts{n}(ll),uniqueKnots{n})
               pos=find(active{n}==ll,1);
              [C,THETA{n}(ll)]=JUMPTERMANI(X{n}(active{n}(pos)-2),Y{n}(active{n
                  }(pos)-2),X{n}(active{n}(pos)-1),Y{n}(active{n}(pos)-1),X{n}(
                  active{n}(pos)),Y{n}(active{n}(pos)),X{n}(active{n}(pos)),Y{n
                  }(active{n}(pos)),X{n}(active{n}(pos)+1),Y{n}(active{n}(pos)
                  +1),X{n}(active{n}(pos)+2),Y{n}(active{n}(pos)+2),collocPts{n
                  }(ll),collocPts{n}(ll),knotVec{n},weight{n},active{n}(pos)-2,
                  active{n}(pos),mu_k,Ajump,Bjump,K,D);
           else
              % C(1,1)=0.5; C(1,2)=0; C(2,1)=0; C(2,2)=0.5;
              THETA{n}(ll)=pi;
           end
       else
            [C,THETA{n}(ll)]=JUMPTERMANI(X{n}(N(n)-1),Y{n}(N(n)-1),X{n}(N(n)),Y{
                n}(N(n)),X{n}(N(n)+1),Y{n}(N(n)+1),X{n}(1),Y{n}(1),X{n}(2),Y{n
                }(2),X{n}(3),Y{n}(3),collocPts{n}(end),collocPts{n}(1),knotVec{n
                },weight{n},active{n}(end),active{n}(1),mu_k,Ajump,Bjump,K,D);
       end
            % H(2*ll-1+pGH:2*ll+pGH,:)=H(2*ll-1+pGH:2*ll+pGH,:)+repmat(C,[1,NT])
                .*ACfinal(2*ll-1+pGH:2*ll+pGH,:);
       % Carry out the integration

       for nn=1:NN
          for i=1:size(elRanges{nn},1)
              if nn==n
                 if (collocPts{n}(ll)>=elRanges{n}(i,1)&& collocPts{n}(ll)<=
                     elRanges{n}(i,2)) || (ll==1 && i==size(elRanges{nn},1))
```

```
                [HW]=SSTANI(collocPtsX{n}(ll),collocPtsY{n}(ll),X{nn}(active
                    {nn}(i)),Y{nn}(active{nn}(i)),X{nn}(active{nn}(i)+1),Y{
                    nn}(active{nn}(i)+1),X{nn}(active{nn}(i)+2),Y{nn}(active
                    {nn}(i)+2),elRanges{nn}(i,1),elRanges{nn}(i,2),collocPts
                    {n}(ll),knotVec{nn},weight{nn},active{nn}(i),Qmat,Lmat,
                    mu_k,collocPtsX{n}(1),collocPtsY{n}(1),i);
                [GW]=LOCINEQIGABEMANI2(collocPtsX{n}(ll),collocPtsY{n}(ll),X
                    {nn}(active{nn}(i)),Y{nn}(active{nn}(i)),X{nn}(active{nn
                    }(i)+1),Y{nn}(active{nn}(i)+1),X{nn}(active{nn}(i)+2),Y{
                    nn}(active{nn}(i)+2),elRanges{n}(i,1),elRanges{n}(i,2),
                    collocPts{n}(ll),knotVec{n},weight{n},active{nn}(i),Qmat
                    ,Amat,mu_k,collocPtsX{n}(1),collocPtsY{n}(1),i);
             else
                [HW,GW]=EXTINEQIGABEMANI(collocPtsX{n}(ll),collocPtsY{n}(ll)
                    ,X{nn}(active{nn}(i)),Y{nn}(active{nn}(i)),X{nn}(active{
                    nn}(i)+1),Y{nn}(active{nn}(i)+1),X{nn}(active{nn}(i)+2),
                    Y{nn}(active{nn}(i)+2),elRanges{nn}(i,1),elRanges{nn}(i
                    ,2),knotVec{nn},weight{nn},active{nn}(i),Qmat,Amat,Lmat,
                    mu_k,collocPtsX{n}(1),collocPtsY{n}(1),i);
             end
          else
              [HW,GW]=EXTINEQIGABEMANI(collocPtsX{n}(ll),collocPtsY{n}(ll),X{
                  nn}(active{nn}(i)),Y{nn}(active{nn}(i)),X{nn}(active{nn}(i)
                  +1),Y{nn}(active{nn}(i)+1),X{nn}(active{nn}(i)+2),Y{nn}(
                  active{nn}(i)+2),elRanges{nn}(i,1),elRanges{nn}(i,2),
                  knotVec{nn},weight{nn},active{nn}(i),Qmat,Amat,Lmat,mu_k,
                  collocPtsX{n}(1),collocPtsY{n}(1),i);
          end

          % Plug the GW and the HW matrices into the general G and H
              matrices.

          for k=1:3
             for j=1:9
                G(3*ll-3+k+pGH,4.5*(active{nn}(i)-1)+j+pG)=G(3*ll-3+k+pGH
                    ,4.5*(active{nn}(i)-1)+j+pG)+GW(k,j);
                if (i==size(elRanges{nn},1) && j>=7)
                       H(3*ll-3+k+pGH,j+pH-6)=H(3*ll-3+k+pGH,j+pH-6)+HW(k,j);

                else
                       H(3*ll-3+k+pGH,3*(active{nn}(i)-1)+j+pH)=H(3*ll-3+k+
                           pGH,3*(active{nn}(i)-1)+j+pH)+HW(k,j);
                end
             end
          end
          if i==n1{nn}(1)
              pG=pG+9*size(elRanges{nn},1);
              pH=pH+3*N(nn);
          end
       end
    end
    if ll==N(n)
        pGH=pGH+3*ll;
    end
 end
end
```

```
%
    ----------------------------------------------------------------------------------------------------


% If the jump term is not in use, apply rigid movement:

for i=1:1:3*NT
    sum1=sum(H(i,1:3:end));
    sum2=sum(H(i,2:3:end));
    sum3=sum(H(i,3:3:end));
    for j=1:3:3*NT
        H(i,j)=H(i,j)-sum1*AC{1}(i,j);
    end
    for j=2:3:3*NT
        H(i,j)=H(i,j)-sum2*AC{1}(i,j);
    end
    for j=3:3:3*NT
        H(i,j)=H(i,j)-sum3*AC{1}(i,j);
    end
end
%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


Hfinal=H/Afinal;
Gfinal=G/Bfinal;
sum(Hfinal')
% Create KODE and DFI as usable arrays.

for k=1:sum(N)
   KODEu(3*k-2)=displacements(6*k-5);
   KODEu(3*k-1)=displacements(6*k-3);
   KODEu(3*k)=displacements(6*k-1);
   DFIu(3*k-2)=displacements(6*k-4);
   DFIu(3*k-1)=displacements(6*k-2);
   DFIu(3*k)=displacements(6*k);
end
KODEu=logical(KODEu);
DFIu=DFIu(KODEu);

for k=1:length(tractions)/2
    KODEv(k)=tractions(2*k-1);
    DFIv(k)=tractions(2*k);
end


% Manage the variables of G (and B)

for n=1:NN
    if n==1
        Thetafinal=THETA{n};
    else
        Thetafinal=[Thetafinal,THETA{n}];
    end
end

for j=1:size(Gfinal,2)
```

```
    Gcolumn(j)=Activefinal(ceil(j/9))+floor(rem(j-1,9)/3);
end

pos=0;
for n=1:NN
    Gcolumn(9*length(active{n})-2+pos:9*length(active{n})+pos)=Gcolumn(9*length
        (active{n})-2+pos:9*length(active{n})+pos)-3*size(elRanges{nn},1);
    pos=9*length(active{n})+pos;
end

Gs=Gcolumn;

for j=10:3:size(Gfinal,2)
    for k=1:3:j-3
        if Gcolumn(j)~=0
            if Gcolumn(k)==Gcolumn(j) && (pi-20*eps)<=Thetafinal(Gcolumn(j)) &&
                Thetafinal(Gcolumn(j))<=(pi+20*eps)
                Gfinal(:,k)=Gfinal(:,j)+Gfinal(:,k);
                Gfinal(:,k+1)=Gfinal(:,j+1)+Gfinal(:,k+1);
                Gfinal(:,k+2)=Gfinal(:,j+2)+Gfinal(:,k+2);
                Gfinal(:,j+2)=0;
                Gfinal(:,j+1)=0;
                Gfinal(:,j)=0;
                Gcolumn(:,j+2)=0;
                Gcolumn(:,j+1)=0;
                Gcolumn(:,j)=0;
            end
       end
    end
end

KODE1v=KODEv;
DFI1v=DFIv;

for j=size(Gcolumn,2):-1:1
    if Gfinal(:,j)==0
        Gfinal(:,j)=[];
        Gcolumn(:,j)=[];
        KODEv(j)=[];
        DFIv(j)=[];
    end
end

KODEv=logical(KODEv);
DFIv=DFIv(KODEv);

% Reorder the columns of the system of equations in accordance with the
% boundary conditions and form system matrix C*x=FI.

Du=Hfinal(:,KODEu);
Cu=Hfinal(:,~KODEu);
Dv=Gfinal(:,KODEv);
Cv=Gfinal(:,~KODEv);
D=[-Du,Dv];
C=[Cu,-Cv];
DFI=[DFIu,DFIv];
```

```
FI=D*DFI';
```

B.4 JUMPTERMANI

```
function [C,THETA]=JUMPTERMANI(X1,Y1,X2,Y2,X3,Y3,X4,Y4,X5,Y5,X6,Y6,COL1,COL2,
    knotVec,weight,active1,active2,mu_k,Ajump,Bjump,K,D)

[~,dR11]=NURBSbasis_matlab(active1,2,COL1-10*eps,knotVec,weight);
[~,dR21]=NURBSbasis_matlab(active1+1,2,COL1-10*eps,knotVec,weight);
[~,dR31]=NURBSbasis_matlab(active1+2,2,COL1-10*eps,knotVec,weight);
[~,dR12]=NURBSbasis_matlab(active2,2,COL2+10*eps,knotVec,weight);
[~,dR22]=NURBSbasis_matlab(active2+1,2,COL2+10*eps,knotVec,weight);
[~,dR32]=NURBSbasis_matlab(active2+2,2,COL2+10*eps,knotVec,weight);

A1=dR11*X1+dR21*X2+dR31*X3;
B1=dR11*Y1+dR21*Y2+dR31*Y3;
MG1=sqrt(A1^2+B1^2);
R11=-A1/MG1;
R21=-B1/MG1;

A2=dR12*X4+dR22*X5+dR32*X6;
B2=dR12*Y4+dR22*Y5+dR32*Y6;
MG2=sqrt(A2^2+B2^2);
R12=A2/MG2;
R22=B2/MG2;

if D==1
%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    z1=R11+mu_k(1)*R21;
    z2=R12+mu_k(1)*R22;

    G1=1/(pi*1i*K)*[conj(z1)/z1,log(z1);log(z1),0];
    G2=1/(pi*1i*K)*[conj(z2)/z2,log(z2);log(z2),0];
    if COL1==3
        G1=1/(pi*1i*K)*[conj(z1)/z1,log(z1)+2*pi*1i;log(z1)+2*pi*1i,0];
    end

    C(1,1)=real(Bjump(1,1)*Ajump(1,1)*G1(1,1)+Bjump(1,1)*Ajump(1,2)*G1(1,2)+
        Bjump(1,2)*Ajump(1,1)*G1(2,1)+Bjump(1,2)*Ajump(1,2)*G1(2,2)-Bjump(1,1)*
        Ajump(1,1)*G2(1,1)-Bjump(1,1)*Ajump(1,2)*G2(1,2)-Bjump(1,2)*Ajump(1,1)*
        G2(2,1)-Bjump(1,2)*Ajump(1,2)*G2(2,2));
    C(1,2)=real(Bjump(1,1)*Ajump(2,1)*G1(1,1)+Bjump(1,1)*Ajump(2,2)*G1(1,2)+
        Bjump(1,2)*Ajump(2,1)*G1(2,1)+Bjump(1,2)*Ajump(2,2)*G1(2,2)-Bjump(1,1)*
        Ajump(2,1)*G2(1,1)-Bjump(1,1)*Ajump(2,2)*G2(1,2)-Bjump(1,2)*Ajump(2,1)*
        G2(2,1)-Bjump(1,2)*Ajump(2,2)*G2(2,2));
    C(2,1)=real(Bjump(2,1)*Ajump(1,1)*G1(1,1)+Bjump(2,1)*Ajump(1,2)*G1(1,2)+
        Bjump(2,2)*Ajump(1,1)*G1(2,1)+Bjump(2,2)*Ajump(1,2)*G1(2,2)-Bjump(2,1)*
        Ajump(1,1)*G2(1,1)-Bjump(2,1)*Ajump(1,2)*G2(1,2)-Bjump(2,2)*Ajump(1,1)*
        G2(2,1)-Bjump(2,2)*Ajump(1,2)*G2(2,2));
    C(2,2)=real(Bjump(2,1)*Ajump(2,1)*G1(1,1)+Bjump(2,1)*Ajump(2,2)*G1(1,2)+
        Bjump(2,2)*Ajump(2,1)*G1(2,1)+Bjump(2,2)*Ajump(2,2)*G1(2,2)-Bjump(2,1)*
```

```
        Ajump(2,1)*G2(1,1)-Bjump(2,1)*Ajump(2,2)*G2(1,2)-Bjump(2,2)*Ajump(2,1)*
        G2(2,1)-Bjump(2,2)*Ajump(2,2)*G2(2,2));
%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

else
    z11=R11+mu_k(1)*R21;
    z12=R12+mu_k(1)*R22;

    z21=R11+mu_k(2)*R21;
    z22=R12+mu_k(2)*R22;

    C(1,1)=real(1/(pi*1i*K(1))*Bjump(1,1)*Ajump(1,1)*log(z11/z12)+1/(pi*1i*K(2)
        )*Bjump(1,2)*Ajump(1,2)*log(z21/z22));
    C(1,2)=real(1/(pi*1i*K(1))*Bjump(1,1)*Ajump(2,1)*log(z11/z12)+1/(pi*1i*K(2)
        )*Bjump(1,2)*Ajump(2,2)*log(z21/z22));
    C(2,1)=real(1/(pi*1i*K(1))*Bjump(2,1)*Ajump(1,1)*log(z11/z12)+1/(pi*1i*K(2)
        )*Bjump(2,2)*Ajump(1,2)*log(z21/z22));
    C(2,2)=real(1/(pi*1i*K(1))*Bjump(2,1)*Ajump(2,1)*log(z11/z12)+1/(pi*1i*K(2)
        )*Bjump(2,2)*Ajump(2,2)*log(z21/z22));

end

THETA=atan2d(A2*-B1-B2*-A1,A2*-A1+B2*-B1)*pi/180;

if THETA<0
    THETA=THETA+2*pi;
end
if THETA<0
    THETA=THETA+2*pi;
end
end
```

B.5 LOCINEQIGABEMANI

```
function [GW]=LOCINEQIGABEMANI2(XP,YP,X1,Y1,X2,Y2,X3,Y3,inicio,final,COL,
    knotVec,weight,active,Qmat,Amat,mu_k,Xfirst,Yfirst,vecpos)

%This subroutine computes the GW matriz when the collocation point is one of
    the nodes of the integration element.
%The coefficients are computed by numerical integration: the non singular
%part is computed using standard Gauss quadrature, the logarithmic part is
%computed using a special quadrature formula.

global NURBS dNURBS
%Data for the Gauss quadrature.

GI=[0.9739065285,-0.9739065285,0.8650633666,-0.8650633666,0.6794095682,
-0.6794095682,0.4333953941,-0.4333953941,0.1488743389,-0.1488743389];
OME=[0.0666713443,0.0666713443,0.1494513491,0.1494513491,0.2190863625,
0.2190863625,0.2692667193,0.2692667193,0.2955242247,0.2955242247];

% Data for the special quadrature.
```

```
GIL=[0.0090426309,0.0539712662,0.1353118246,0.2470524162,0.3802125396,
0.5237923179,0.6657752055,0.7941904160,0.8981610912,0.9688479887];
OMEL=[0.1209551319,0.1863635425,0.1956608732,0.1735771421,0.1356956729,
0.0936467585,0.0557877273,0.0271598109,0.0095151826,0.0016381576];

if COL<inicio
    COL=final;
end

if COL~=inicio && COL~=final
    NODO=2;
elseif COL==inicio
     NODO=1;
elseif COL==final
    NODO=3;
end

GW=zeros(3,9);
JAC=(final-inicio)/2;

% Transformation to complex plane

zo1=XP+YP*mu_k(1);
zo2=XP+YP*mu_k(2);
zo3=XP+YP*mu_k(3);

for i=1:10

    % Compute shape functions for numerical integration.

    if XP==Xfirst && YP==Yfirst
        [F1,dR1]=NURBSbasis_matlab(active,2,(GI(i)*(final-inicio)+final+inicio)
            /2,knotVec,weight);
        [F2,dR2]=NURBSbasis_matlab(active+1,2,(GI(i)*(final-inicio)+final+
            inicio)/2,knotVec,weight);
        [F3,dR3]=NURBSbasis_matlab(active+2,2,(GI(i)*(final-inicio)+final+
            inicio)/2,knotVec,weight);
        NURBS(active,10*vecpos-10+i)=F1;
        NURBS(active+1,10*vecpos-10+i)=F2;
        NURBS(active+2,10*vecpos-10+i)=F3;
        dNURBS(active,10*vecpos-10+i)=dR1;
        dNURBS(active+1,10*vecpos-10+i)=dR2;
        dNURBS(active+2,10*vecpos-10+i)=dR3;
    else
        F1=NURBS(active,10*vecpos-10+i);
        F2=NURBS(active+1,10*vecpos-10+i);
        F3=NURBS(active+2,10*vecpos-10+i);
        dR1=dNURBS(active,10*vecpos-10+i);
        dR2=dNURBS(active+1,10*vecpos-10+i);
        dR3=dNURBS(active+2,10*vecpos-10+i);
    end

    XCO=X1*F1+X2*F2+X3*F3;
    YCO=Y1*F1+Y2*F2+Y3*F3;

    % Transformation to complex plane
```

```
z1=XCO+YCO*mu_k(1);
z2=XCO+YCO*mu_k(2);
z3=XCO+YCO*mu_k(3);

% Jacobian

A=dR1*X1+dR2*X2+dR3*X3;
B=dR1*Y1+dR2*Y2+dR3*Y3;
XJA=sqrt(A^2+B^2);

%Compute GW coefficients

 if NODO==1

    [FL11,dR11]=NURBSbasis_matlab(active,2,GIL(i)*(final-inicio)+inicio,
        knotVec,weight);
    [FL12,dR12]=NURBSbasis_matlab(active+1,2,GIL(i)*(final-inicio)+inicio,
        knotVec,weight);
    [FL13,dR13]=NURBSbasis_matlab(active+2,2,GIL(i)*(final-inicio)+inicio,
        knotVec,weight);

    AL=dR11*X1+dR12*X2+dR13*X3;
    BL=dR11*Y1+dR12*Y2+dR13*Y3;
    XJA2=sqrt(AL^2+BL^2);

    L11=+JAC*XJA2*2*OMEL(i)/pi*real(Amat(1,1)*Qmat(1,1)+Amat(1,2)*Qmat(2,1)
        +Amat(1,3)*Qmat(3,1));
    L12=+JAC*XJA2*2*OMEL(i)/pi*real(Amat(2,1)*Qmat(1,1)+Amat(2,2)*Qmat(2,1)
        +Amat(2,3)*Qmat(3,1));
    L13=+JAC*XJA2*2*OMEL(i)/pi*real(Amat(3,1)*Qmat(1,1)+Amat(3,2)*Qmat(2,1)
        +Amat(3,3)*Qmat(3,1));
    L21=+JAC*XJA2*2*OMEL(i)/pi*real(Amat(1,1)*Qmat(1,2)+Amat(1,2)*Qmat(2,2)
        +Amat(1,3)*Qmat(3,2));
    L22=+JAC*XJA2*2*OMEL(i)/pi*real(Amat(2,1)*Qmat(1,2)+Amat(2,2)*Qmat(2,2)
        +Amat(2,3)*Qmat(3,2));
    L23=+JAC*XJA2*2*OMEL(i)/pi*real(Amat(3,1)*Qmat(1,2)+Amat(3,2)*Qmat(2,2)
        +Amat(3,3)*Qmat(3,2));
    L31=+JAC*XJA2*2*OMEL(i)/pi*real(Amat(1,1)*Qmat(1,3)+Amat(1,2)*Qmat(2,3)
        +Amat(1,3)*Qmat(3,3));
    L32=+JAC*XJA2*2*OMEL(i)/pi*real(Amat(2,1)*Qmat(1,3)+Amat(2,2)*Qmat(2,3)
        +Amat(2,3)*Qmat(3,3));
    L33=+JAC*XJA2*2*OMEL(i)/pi*real(Amat(3,1)*Qmat(1,3)+Amat(3,2)*Qmat(2,3)
        +Amat(3,3)*Qmat(3,3));

    NU=(GI(i)+1)/2;

    S11=-JAC*XJA*OME(i)/pi*real(Amat(1,1)*Qmat(1,1)*log((z1-zo1)/NU)+Amat
        (1,2)*Qmat(2,1)*log((z2-zo2)/NU)+Amat(1,3)*Qmat(3,1)*log((z3-zo3)/
        NU));
    S12=-JAC*XJA*OME(i)/pi*real(Amat(2,1)*Qmat(1,1)*log((z1-zo1)/NU)+Amat
        (2,2)*Qmat(2,1)*log((z2-zo2)/NU)+Amat(2,3)*Qmat(3,1)*log((z3-zo3)/
        NU));
    S13=-JAC*XJA*OME(i)/pi*real(Amat(3,1)*Qmat(1,1)*log((z1-zo1)/NU)+Amat
        (3,2)*Qmat(2,1)*log((z2-zo2)/NU)+Amat(3,3)*Qmat(3,1)*log((z3-zo3)/
        NU));
```

```
        S21=-JAC*XJA*OME(i)/pi*real(Amat(1,1)*Qmat(1,2)*log((z1-zo1)/NU)+Amat
            (1,2)*Qmat(2,2)*log((z2-zo2)/NU)+Amat(1,3)*Qmat(3,2)*log((z3-zo3)/
            NU));
        S22=-JAC*XJA*OME(i)/pi*real(Amat(2,1)*Qmat(1,2)*log((z1-zo1)/NU)+Amat
            (2,2)*Qmat(2,2)*log((z2-zo2)/NU)+Amat(2,3)*Qmat(3,2)*log((z3-zo3)/
            NU));
        S23=-JAC*XJA*OME(i)/pi*real(Amat(3,1)*Qmat(1,2)*log((z1-zo1)/NU)+Amat
            (3,2)*Qmat(2,2)*log((z2-zo2)/NU)+Amat(3,3)*Qmat(3,2)*log((z3-zo3)/
            NU));
        S31=-JAC*XJA*OME(i)/pi*real(Amat(1,1)*Qmat(1,3)*log((z1-zo1)/NU)+Amat
            (1,2)*Qmat(2,3)*log((z2-zo2)/NU)+Amat(1,3)*Qmat(3,3)*log((z3-zo3)/
            NU));
        S32=-JAC*XJA*OME(i)/pi*real(Amat(2,1)*Qmat(1,3)*log((z1-zo1)/NU)+Amat
            (2,2)*Qmat(2,3)*log((z2-zo2)/NU)+Amat(2,3)*Qmat(3,3)*log((z3-zo3)/
            NU));
        S33=-JAC*XJA*OME(i)/pi*real(Amat(3,1)*Qmat(1,3)*log((z1-zo1)/NU)+Amat
            (3,2)*Qmat(2,3)*log((z2-zo2)/NU)+Amat(3,3)*Qmat(3,3)*log((z3-zo3)/
            NU));
        GW(1,1)=GW(1,1)+S11*F1+L11*FL11;
        GW(1,2)=GW(1,2)+S12*F1+L12*FL11;
        GW(1,3)=GW(1,3)+S13*F1+L13*FL11;
        GW(1,4)=GW(1,4)+S11*F2+L11*FL12;
        GW(1,5)=GW(1,5)+S12*F2+L12*FL12;
        GW(1,6)=GW(1,6)+S13*F2+L13*FL12;
        GW(1,7)=GW(1,7)+S11*F3+L11*FL13;
        GW(1,8)=GW(1,8)+S12*F3+L12*FL13;
        GW(1,9)=GW(1,9)+S13*F3+L13*FL13;
        GW(2,1)=GW(2,1)+S21*F1+L21*FL11;
        GW(2,2)=GW(2,2)+S22*F1+L22*FL11;
        GW(2,3)=GW(2,3)+S23*F1+L23*FL11;
        GW(2,4)=GW(2,4)+S21*F2+L21*FL12;
        GW(2,5)=GW(2,5)+S22*F2+L22*FL12;
        GW(2,6)=GW(2,6)+S23*F2+L23*FL12;
        GW(2,7)=GW(2,7)+S21*F3+L21*FL13;
        GW(2,8)=GW(2,8)+S22*F3+L22*FL13;
        GW(2,9)=GW(2,9)+S23*F3+L23*FL13;
        GW(3,1)=GW(3,1)+S31*F1+L31*FL11;
        GW(3,2)=GW(3,2)+S32*F1+L32*FL11;
        GW(3,3)=GW(3,3)+S33*F1+L33*FL11;
        GW(3,4)=GW(3,4)+S31*F2+L31*FL12;
        GW(3,5)=GW(3,5)+S32*F2+L32*FL12;
        GW(3,6)=GW(3,6)+S33*F2+L33*FL12;
        GW(3,7)=GW(3,7)+S31*F3+L31*FL13;
        GW(3,8)=GW(3,8)+S32*F3+L32*FL13;
        GW(3,9)=GW(3,9)+S33*F3+L33*FL13;
    end

    if NODO==2

        [FA1,dRA1]=NURBSbasis_matlab(active,2,GIL(i)*(inicio-COL)+COL,knotVec,
            weight);
        [FA2,dRA2]=NURBSbasis_matlab(active+1,2,GIL(i)*(inicio-COL)+COL,
            knotVec,weight);
        [FA3,dRA3]=NURBSbasis_matlab(active+2,2,GIL(i)*(inicio-COL)+COL,
            knotVec,weight);
        [FB1,dRB1]=NURBSbasis_matlab(active,2,GIL(i)*(final-COL)+COL,knotVec,
            weight);
```

```
        [FB2,dRB2]=NURBSbasis_matlab(active+1,2,GIL(i)*(final-COL)+COL,knotVec,
            weight);
        [FB3,dRB3]=NURBSbasis_matlab(active+2,2,GIL(i)*(final-COL)+COL,knotVec,
            weight);

        Aa=dRA1*X1+dRA2*X2+dRA3*X3;
        Ba=dRA1*Y1+dRA2*Y2+dRA3*Y3;
        XJAa=sqrt(Aa^2+Ba^2);
        Ab=dRB1*X1+dRB2*X2+dRB3*X3;
        Bb=dRB1*Y1+dRB2*Y2+dRB3*Y3;
        XJAb=sqrt(Ab^2+Bb^2);

        L11a=+JAC*XJAa*OMEL(i)/pi*real(Amat(1,1)*Qmat(1,1)+Amat(1,2)*Qmat(2,1)+
            Amat(1,3)*Qmat(3,1));
        L12a=+JAC*XJAa*OMEL(i)/pi*real(Amat(2,1)*Qmat(1,1)+Amat(2,2)*Qmat(2,1)+
            Amat(2,3)*Qmat(3,1));
        L13a=+JAC*XJAa*OMEL(i)/pi*real(Amat(3,1)*Qmat(1,1)+Amat(3,2)*Qmat(2,1)+
            Amat(3,3)*Qmat(3,1));
        L21a=+JAC*XJAa*OMEL(i)/pi*real(Amat(1,1)*Qmat(1,2)+Amat(1,2)*Qmat(2,2)+
            Amat(1,3)*Qmat(3,2));
        L22a=+JAC*XJAa*OMEL(i)/pi*real(Amat(2,1)*Qmat(1,2)+Amat(2,2)*Qmat(2,2)+
            Amat(2,3)*Qmat(3,2));
        L23a=+JAC*XJAa*OMEL(i)/pi*real(Amat(3,1)*Qmat(1,2)+Amat(3,2)*Qmat(2,2)+
            Amat(3,3)*Qmat(3,2));
        L31a=+JAC*XJAa*OMEL(i)/pi*real(Amat(1,1)*Qmat(1,3)+Amat(1,2)*Qmat(2,3)+
            Amat(1,3)*Qmat(3,3));
        L32a=+JAC*XJAa*OMEL(i)/pi*real(Amat(2,1)*Qmat(1,3)+Amat(2,2)*Qmat(2,3)+
            Amat(2,3)*Qmat(3,3));
        L33a=+JAC*XJAa*OMEL(i)/pi*real(Amat(3,1)*Qmat(1,3)+Amat(3,2)*Qmat(2,3)+
            Amat(3,3)*Qmat(3,3));

        L11b=+JAC*XJAb*OMEL(i)/pi*real(Amat(1,1)*Qmat(1,1)+Amat(1,2)*Qmat(2,1)+
            Amat(1,3)*Qmat(3,1));
        L12b=+JAC*XJAb*OMEL(i)/pi*real(Amat(2,1)*Qmat(1,1)+Amat(2,2)*Qmat(2,1)+
            Amat(2,3)*Qmat(3,1));
        L13b=+JAC*XJAb*OMEL(i)/pi*real(Amat(3,1)*Qmat(1,1)+Amat(3,2)*Qmat(2,1)+
            Amat(3,3)*Qmat(3,1));
        L21b=+JAC*XJAb*OMEL(i)/pi*real(Amat(1,1)*Qmat(1,2)+Amat(1,2)*Qmat(2,2)+
            Amat(1,3)*Qmat(3,2));
        L22b=+JAC*XJAb*OMEL(i)/pi*real(Amat(2,1)*Qmat(1,2)+Amat(2,2)*Qmat(2,2)+
            Amat(2,3)*Qmat(3,2));
        L23b=+JAC*XJAb*OMEL(i)/pi*real(Amat(3,1)*Qmat(1,2)+Amat(3,2)*Qmat(2,2)+
            Amat(3,3)*Qmat(3,2));
        L31b=+JAC*XJAb*OMEL(i)/pi*real(Amat(1,1)*Qmat(1,3)+Amat(1,2)*Qmat(2,3)+
            Amat(1,3)*Qmat(3,3));
        L32b=+JAC*XJAb*OMEL(i)/pi*real(Amat(2,1)*Qmat(1,3)+Amat(2,2)*Qmat(2,3)+
            Amat(2,3)*Qmat(3,3));
        L33b=+JAC*XJAb*OMEL(i)/pi*real(Amat(3,1)*Qmat(1,3)+Amat(3,2)*Qmat(2,3)+
            Amat(3,3)*Qmat(3,3));

        S11=-JAC*XJA*OME(i)/pi*real(Amat(1,1)*Qmat(1,1)*log((z1-zo1)/abs(GI(i))
            )+Amat(1,2)*Qmat(2,1)*log((z2-zo2)/abs(GI(i)))+Amat(1,3)*Qmat(3,1)*
            log((z3-zo3)/abs(GI(i))));
        S12=-JAC*XJA*OME(i)/pi*real(Amat(2,1)*Qmat(1,1)*log((z1-zo1)/abs(GI(i))
            )+Amat(2,2)*Qmat(2,1)*log((z2-zo2)/abs(GI(i)))+Amat(2,3)*Qmat(3,1)*
            log((z3-zo3)/abs(GI(i))));
```

```
        S13=-JAC*XJA*OME(i)/pi*real(Amat(3,1)*Qmat(1,1)*log((z1-zo1)/abs(GI(i))
            )+Amat(3,2)*Qmat(2,1)*log((z2-zo2)/abs(GI(i)))+Amat(3,3)*Qmat(3,1)*
            log((z3-zo3)/abs(GI(i))));
        S21=-JAC*XJA*OME(i)/pi*real(Amat(1,1)*Qmat(1,2)*log((z1-zo1)/abs(GI(i))
            )+Amat(1,2)*Qmat(2,2)*log((z2-zo2)/abs(GI(i)))+Amat(1,3)*Qmat(3,2)*
            log((z3-zo3)/abs(GI(i))));
        S22=-JAC*XJA*OME(i)/pi*real(Amat(2,1)*Qmat(1,2)*log((z1-zo1)/abs(GI(i))
            )+Amat(2,2)*Qmat(2,2)*log((z2-zo2)/abs(GI(i)))+Amat(2,3)*Qmat(3,2)*
            log((z3-zo3)/abs(GI(i))));
        S23=-JAC*XJA*OME(i)/pi*real(Amat(3,1)*Qmat(1,2)*log((z1-zo1)/abs(GI(i))
            )+Amat(3,2)*Qmat(2,2)*log((z2-zo2)/abs(GI(i)))+Amat(3,3)*Qmat(3,2)*
            log((z3-zo3)/abs(GI(i))));
        S31=-JAC*XJA*OME(i)/pi*real(Amat(1,1)*Qmat(1,3)*log((z1-zo1)/abs(GI(i))
            )+Amat(1,2)*Qmat(2,3)*log((z2-zo2)/abs(GI(i)))+Amat(1,3)*Qmat(3,3)*
            log((z3-zo3)/abs(GI(i))));
        S32=-JAC*XJA*OME(i)/pi*real(Amat(2,1)*Qmat(1,3)*log((z1-zo1)/abs(GI(i))
            )+Amat(2,2)*Qmat(2,3)*log((z2-zo2)/abs(GI(i)))+Amat(2,3)*Qmat(3,3)*
            log((z3-zo3)/abs(GI(i))));
        S33=-JAC*XJA*OME(i)/pi*real(Amat(3,1)*Qmat(1,3)*log((z1-zo1)/abs(GI(i))
            )+Amat(3,2)*Qmat(2,3)*log((z2-zo2)/abs(GI(i)))+Amat(3,3)*Qmat(3,3)*
            log((z3-zo3)/abs(GI(i))));

        GW(1,1)=GW(1,1)+L11a*FA1+L11b*FB1+S11*F1;
        GW(1,2)=GW(1,2)+L12a*FA1+L12b*FB1+S12*F1;
        GW(1,3)=GW(1,3)+L13a*FA1+L13b*FB1+S13*F1;
        GW(1,4)=GW(1,4)+L11a*FA2+L11b*FB2+S11*F2;
        GW(1,5)=GW(1,5)+L12a*FA2+L12b*FB2+S12*F2;
        GW(1,6)=GW(1,6)+L13a*FA2+L13b*FB2+S13*F2;
        GW(1,7)=GW(1,7)+L11a*FA3+L11b*FB3+S11*F3;
        GW(1,8)=GW(1,8)+L12a*FA3+L12b*FB3+S12*F3;
        GW(1,9)=GW(1,9)+L13a*FA3+L13b*FB3+S13*F3;
        GW(2,1)=GW(2,1)+L21a*FA1+L21b*FB1+S21*F1;
        GW(2,2)=GW(2,2)+L22a*FA1+L22b*FB1+S22*F1;
        GW(2,3)=GW(2,3)+L23a*FA1+L23b*FB1+S23*F1;
        GW(2,4)=GW(2,4)+L21a*FA2+L21b*FB2+S21*F2;
        GW(2,5)=GW(2,5)+L22a*FA2+L22b*FB2+S22*F2;
        GW(2,6)=GW(2,6)+L23a*FA2+L23b*FB2+S23*F2;
        GW(2,7)=GW(2,7)+L21a*FA3+L21b*FB3+S21*F3;
        GW(2,8)=GW(2,8)+L22a*FA3+L22b*FB3+S22*F3;
        GW(2,9)=GW(2,9)+L23a*FA3+L23b*FB3+S23*F3;
        GW(3,1)=GW(3,1)+L31a*FA1+L31b*FB1+S31*F1;
        GW(3,2)=GW(3,2)+L32a*FA1+L32b*FB1+S32*F1;
        GW(3,3)=GW(3,3)+L33a*FA1+L33b*FB1+S33*F1;
        GW(3,4)=GW(3,4)+L31a*FA2+L31b*FB2+S31*F2;
        GW(3,5)=GW(3,5)+L32a*FA2+L32b*FB2+S32*F2;
        GW(3,6)=GW(3,6)+L33a*FA2+L33b*FB2+S33*F2;
        GW(3,7)=GW(3,7)+L31a*FA3+L31b*FB3+S31*F3;
        GW(3,8)=GW(3,8)+L32a*FA3+L32b*FB3+S32*F3;
        GW(3,9)=GW(3,9)+L33a*FA3+L33b*FB3+S33*F3;
    end

    if NODO==3

        [FL31,dR31]=NURBSbasis_matlab(active,2,GIL(i)*(inicio-final)+final,
            knotVec,weight);
        [FL32,dR32]=NURBSbasis_matlab(active+1,2,GIL(i)*(inicio-final)+final,
            knotVec,weight);
```

```
[FL33,dR33]=NURBSbasis_matlab(active+2,2,GIL(i)*(inicio-final)+final,
    knotVec,weight);

AL=dR31*X1+dR32*X2+dR33*X3;
BL=dR31*Y1+dR32*Y2+dR33*Y3;
XJA2=sqrt(AL^2+BL^2);

L11=+JAC*XJA2*2*OMEL(i)/pi*real(Amat(1,1)*Qmat(1,1)+Amat(1,2)*Qmat(2,1)
    +Amat(1,3)*Qmat(3,1));
L12=+JAC*XJA2*2*OMEL(i)/pi*real(Amat(2,1)*Qmat(1,1)+Amat(2,2)*Qmat(2,1)
    +Amat(2,3)*Qmat(3,1));
L13=+JAC*XJA2*2*OMEL(i)/pi*real(Amat(3,1)*Qmat(1,1)+Amat(3,2)*Qmat(2,1)
    +Amat(3,3)*Qmat(3,1));
L21=+JAC*XJA2*2*OMEL(i)/pi*real(Amat(1,1)*Qmat(1,2)+Amat(1,2)*Qmat(2,2)
    +Amat(1,3)*Qmat(3,2));
L22=+JAC*XJA2*2*OMEL(i)/pi*real(Amat(2,1)*Qmat(1,2)+Amat(2,2)*Qmat(2,2)
    +Amat(2,3)*Qmat(3,2));
L23=+JAC*XJA2*2*OMEL(i)/pi*real(Amat(3,1)*Qmat(1,2)+Amat(3,2)*Qmat(2,2)
    +Amat(3,3)*Qmat(3,2));
L31=+JAC*XJA2*2*OMEL(i)/pi*real(Amat(1,1)*Qmat(1,3)+Amat(1,2)*Qmat(2,3)
    +Amat(1,3)*Qmat(3,3));
L32=+JAC*XJA2*2*OMEL(i)/pi*real(Amat(2,1)*Qmat(1,3)+Amat(2,2)*Qmat(2,3)
    +Amat(2,3)*Qmat(3,3));
L33=+JAC*XJA2*2*OMEL(i)/pi*real(Amat(3,1)*Qmat(1,3)+Amat(3,2)*Qmat(2,3)
    +Amat(3,3)*Qmat(3,3));

NU=(1-GI(i))/2;
S11=-JAC*XJA*OME(i)/pi*real(Amat(1,1)*Qmat(1,1)*log((z1-zo1)/NU)+Amat
    (1,2)*Qmat(2,1)*log((z2-zo2)/NU)+Amat(1,3)*Qmat(3,1)*log((z3-zo3)/
    NU));
S12=-JAC*XJA*OME(i)/pi*real(Amat(2,1)*Qmat(1,1)*log((z1-zo1)/NU)+Amat
    (2,2)*Qmat(2,1)*log((z2-zo2)/NU)+Amat(2,3)*Qmat(3,1)*log((z3-zo3)/
    NU));
S13=-JAC*XJA*OME(i)/pi*real(Amat(3,1)*Qmat(1,1)*log((z1-zo1)/NU)+Amat
    (3,2)*Qmat(2,1)*log((z2-zo2)/NU)+Amat(3,3)*Qmat(3,1)*log((z3-zo3)/
    NU));
S21=-JAC*XJA*OME(i)/pi*real(Amat(1,1)*Qmat(1,2)*log((z1-zo1)/NU)+Amat
    (1,2)*Qmat(2,2)*log((z2-zo2)/NU)+Amat(1,3)*Qmat(3,2)*log((z3-zo3)/
    NU));
S22=-JAC*XJA*OME(i)/pi*real(Amat(2,1)*Qmat(1,2)*log((z1-zo1)/NU)+Amat
    (2,2)*Qmat(2,2)*log((z2-zo2)/NU)+Amat(2,3)*Qmat(3,2)*log((z3-zo3)/
    NU));
S23=-JAC*XJA*OME(i)/pi*real(Amat(3,1)*Qmat(1,2)*log((z1-zo1)/NU)+Amat
    (3,2)*Qmat(2,2)*log((z2-zo2)/NU)+Amat(3,3)*Qmat(3,2)*log((z3-zo3)/
    NU));
S31=-JAC*XJA*OME(i)/pi*real(Amat(1,1)*Qmat(1,3)*log((z1-zo1)/NU)+Amat
    (1,2)*Qmat(2,3)*log((z2-zo2)/NU)+Amat(1,3)*Qmat(3,3)*log((z3-zo3)/
    NU));
S32=-JAC*XJA*OME(i)/pi*real(Amat(2,1)*Qmat(1,3)*log((z1-zo1)/NU)+Amat
    (2,2)*Qmat(2,3)*log((z2-zo2)/NU)+Amat(2,3)*Qmat(3,3)*log((z3-zo3)/
    NU));
S33=-JAC*XJA*OME(i)/pi*real(Amat(3,1)*Qmat(1,3)*log((z1-zo1)/NU)+Amat
    (3,2)*Qmat(2,3)*log((z2-zo2)/NU)+Amat(3,3)*Qmat(3,3)*log((z3-zo3)/
    NU));

GW(1,1)=GW(1,1)+S11*F1+L11*FL31;
GW(1,2)=GW(1,2)+S12*F1+L12*FL31;
```

```
        GW(1,3)=GW(1,3)+S13*F1+L13*FL31;
        GW(1,4)=GW(1,4)+S11*F2+L11*FL32;
        GW(1,5)=GW(1,5)+S12*F2+L12*FL32;
        GW(1,6)=GW(1,6)+S13*F2+L13*FL32;
        GW(1,7)=GW(1,7)+S11*F3+L11*FL33;
        GW(1,8)=GW(1,8)+S12*F3+L12*FL33;
        GW(1,9)=GW(1,9)+S13*F3+L13*FL33;
        GW(2,1)=GW(2,1)+S21*F1+L21*FL31;
        GW(2,2)=GW(2,2)+S22*F1+L22*FL31;
        GW(2,3)=GW(2,3)+S23*F1+L23*FL31;
        GW(2,4)=GW(2,4)+S21*F2+L21*FL32;
        GW(2,5)=GW(2,5)+S22*F2+L22*FL32;
        GW(2,6)=GW(2,6)+S23*F2+L23*FL32;
        GW(2,7)=GW(2,7)+S21*F3+L21*FL33;
        GW(2,8)=GW(2,8)+S22*F3+L22*FL33;
        GW(2,9)=GW(2,9)+S23*F3+L23*FL33;
        GW(3,1)=GW(3,1)+S31*F1+L31*FL31;
        GW(3,2)=GW(3,2)+S32*F1+L32*FL31;
        GW(3,3)=GW(3,3)+S33*F1+L33*FL31;
        GW(3,4)=GW(3,4)+S31*F2+L31*FL32;
        GW(3,5)=GW(3,5)+S32*F2+L32*FL32;
        GW(3,6)=GW(3,6)+S33*F2+L33*FL32;
        GW(3,7)=GW(3,7)+S31*F3+L31*FL33;
        GW(3,8)=GW(3,8)+S32*F3+L32*FL33;
        GW(3,9)=GW(3,9)+S33*F3+L33*FL33;
    end
end
end
```

B.6 EXTINEQIGABEMANI

```
function [HW,GW]=EXTINEQIGABEMANI(XP,YP,X1,Y1,X2,Y2,X3,Y3,inicio,final,knotVec,
    weight,active,Qmat,Amat,Lmat,mu_k,Xfirst,Yfirst,vecpos)

% This subroutine computes the HW and GW matrices that relate a node (XP,YP)
    with a boundary element using Gauss quadrature.

%RA=RADIUS.
%RD1,RD2,RDN=RADIUS DERIVATIVES.
%ETA1,ETA2=COMPONENTS OF THE UNIT NORMAL TO THE ELEMENT.
%XCO,YCO=INTEGRATION POINT ALONG THE ELEMENT.
%XJA=JACOBIAN.

global NURBS dNURBS

GI=[0.9739065285,-0.9739065285,0.8650633665,-0.8650633665,0.6794095683,
-0.6794095683,0.4333953941,-0.4333953941,0.1488743389,-0.1488743389];
OME=[0.0666713443,0.0666713443,0.1494513491,0.1494513491,0.2190863625,
0.2190863625,0.2692667193,0.2692667193,0.2955242247,0.2955242247];
HW=zeros(3,9);
GW=zeros(3,9);
JAC=2/(final-inicio);

% Transformation to complex plane
```

```
zo1=XP+YP*mu_k(1);
zo2=XP+YP*mu_k(2);
zo3=XP+YP*mu_k(3);

for i=1:10

    %Compute the values of the shape functions at the integration points.
    if XP==Xfirst && YP==Yfirst
        [F1,dR1]=NURBSbasis_matlab(active,2,(GI(i)*(final-inicio)+final+inicio)
            /2,knotVec,weight);
        [F2,dR2]=NURBSbasis_matlab(active+1,2,(GI(i)*(final-inicio)+final+
            inicio)/2,knotVec,weight);
        [F3,dR3]=NURBSbasis_matlab(active+2,2,(GI(i)*(final-inicio)+final+
            inicio)/2,knotVec,weight);
        NURBS(active,10*vecpos-10+i)=F1;
        NURBS(active+1,10*vecpos-10+i)=F2;
        NURBS(active+2,10*vecpos-10+i)=F3;
        dNURBS(active,10*vecpos-10+i)=dR1;
        dNURBS(active+1,10*vecpos-10+i)=dR2;
        dNURBS(active+2,10*vecpos-10+i)=dR3;
    else
        F1=NURBS(active,10*vecpos-10+i);
        F2=NURBS(active+1,10*vecpos-10+i);
        F3=NURBS(active+2,10*vecpos-10+i);
        dR1=dNURBS(active,10*vecpos-10+i);
        dR2=dNURBS(active+1,10*vecpos-10+i);
        dR3=dNURBS(active+2,10*vecpos-10+i);
    end

    %Compute geometrical properties at the integration points.

    XCO=X1*F1+X2*F2+X3*F3;
    YCO=Y1*F1+Y2*F2+Y3*F3;
    A=dR1*X1+dR2*X2+dR3*X3;
    B=dR1*Y1+dR2*Y2+dR3*Y3;
    XJA=sqrt(A^2+B^2);
    ETA1=B/XJA;
    ETA2=-A/XJA;

    % Transformation to complex plane

    z1=XCO+YCO*mu_k(1);
    z2=XCO+YCO*mu_k(2);
    z3=XCO+YCO*mu_k(3);

    %Compute GW and HW matrices.

%     GW(1,1)=GW(1,1)-OME(i)*XJA*F1/pi*real(Amat(1,1)*Qmat(1,1)*log(z1-zo1)+
    Amat(1,2)*Qmat(2,1)*log(z2-zo2));
%     GW(1,2)=GW(1,2)-OME(i)*XJA*F1/pi*real(Amat(2,1)*Qmat(1,1)*log(z1-zo1)+
    Amat(2,2)*Qmat(2,1)*log(z2-zo2));
%     GW(2,1)=GW(2,1)-OME(i)*XJA*F1/pi*real(Amat(1,1)*Qmat(1,2)*log(z1-zo1)+
    Amat(1,2)*Qmat(2,2)*log(z2-zo2));
%     GW(2,2)=GW(2,2)-OME(i)*XJA*F1/pi*real(Amat(2,1)*Qmat(1,2)*log(z1-zo1)+
    Amat(2,2)*Qmat(2,2)*log(z2-zo2));
```

```
%     HW(1,1)=HW(1,1)+OME(i)*XJA*F1/pi*real(Lmat(1,1)*Qmat(1,1)/(z1-zo1)*(mu_k
    (1)*ETA1-ETA2)+Lmat(1,2)*Qmat(2,1)/(z2-zo2)*(mu_k(2)*ETA1-ETA2));
%     HW(1,2)=HW(1,2)+OME(i)*XJA*F1/pi*real(Lmat(2,1)*Qmat(1,1)/(z1-zo1)*(mu_k
    (1)*ETA1-ETA2)+Lmat(2,2)*Qmat(2,1)/(z2-zo2)*(mu_k(2)*ETA1-ETA2));
%     HW(2,1)=HW(2,1)+OME(i)*XJA*F1/pi*real(Lmat(1,1)*Qmat(1,2)/(z1-zo1)*(mu_k
    (1)*ETA1-ETA2)+Lmat(1,2)*Qmat(2,2)/(z2-zo2)*(mu_k(2)*ETA1-ETA2));
%     HW(2,2)=HW(2,2)+OME(i)*XJA*F1/pi*real(Lmat(2,1)*Qmat(1,2)/(z1-zo1)*(mu_k
    (1)*ETA1-ETA2)+Lmat(2,2)*Qmat(2,2)/(z2-zo2)*(mu_k(2)*ETA1-ETA2));
%     GW(1,3)=GW(1,3)-OME(i)*XJA*F2/pi*real(Amat(1,1)*Qmat(1,1)*log(z1-zo1)+
    Amat(1,2)*Qmat(2,1)*log(z2-zo2));
%     GW(1,4)=GW(1,4)-OME(i)*XJA*F2/pi*real(Amat(2,1)*Qmat(1,1)*log(z1-zo1)+
    Amat(2,2)*Qmat(2,1)*log(z2-zo2));
%     GW(2,3)=GW(2,3)-OME(i)*XJA*F2/pi*real(Amat(1,1)*Qmat(1,2)*log(z1-zo1)+
    Amat(1,2)*Qmat(2,2)*log(z2-zo2));
%     GW(2,4)=GW(2,4)-OME(i)*XJA*F2/pi*real(Amat(2,1)*Qmat(1,2)*log(z1-zo1)+
    Amat(2,2)*Qmat(2,2)*log(z2-zo2));
%     HW(1,3)=HW(1,3)+OME(i)*XJA*F2/pi*real(Lmat(1,1)*Qmat(1,1)/(z1-zo1)*(mu_k
    (1)*ETA1-ETA2)+Lmat(1,2)*Qmat(2,1)/(z2-zo2)*(mu_k(2)*ETA1-ETA2));
%     HW(1,4)=HW(1,4)+OME(i)*XJA*F2/pi*real(Lmat(2,1)*Qmat(1,1)/(z1-zo1)*(mu_k
    (1)*ETA1-ETA2)+Lmat(2,2)*Qmat(2,1)/(z2-zo2)*(mu_k(2)*ETA1-ETA2));
%     HW(2,3)=HW(2,3)+OME(i)*XJA*F2/pi*real(Lmat(1,1)*Qmat(1,2)/(z1-zo1)*(mu_k
    (1)*ETA1-ETA2)+Lmat(1,2)*Qmat(2,2)/(z2-zo2)*(mu_k(2)*ETA1-ETA2));
%     HW(2,4)=HW(2,4)+OME(i)*XJA*F2/pi*real(Lmat(2,1)*Qmat(1,2)/(z1-zo1)*(mu_k
    (1)*ETA1-ETA2)+Lmat(2,2)*Qmat(2,2)/(z2-zo2)*(mu_k(2)*ETA1-ETA2));
%     GW(1,5)=GW(1,5)-OME(i)*XJA*F3/pi*real(Amat(1,1)*Qmat(1,1)*log(z1-zo1)+
    Amat(1,2)*Qmat(2,1)*log(z2-zo2));
%     GW(1,6)=GW(1,6)-OME(i)*XJA*F3/pi*real(Amat(2,1)*Qmat(1,1)*log(z1-zo1)+
    Amat(2,2)*Qmat(2,1)*log(z2-zo2));
%     GW(2,5)=GW(2,5)-OME(i)*XJA*F3/pi*real(Amat(1,1)*Qmat(1,2)*log(z1-zo1)+
    Amat(1,2)*Qmat(2,2)*log(z2-zo2));
%     GW(2,6)=GW(2,6)-OME(i)*XJA*F3/pi*real(Amat(2,1)*Qmat(1,2)*log(z1-zo1)+
    Amat(2,2)*Qmat(2,2)*log(z2-zo2));
%     HW(1,5)=HW(1,5)+OME(i)*XJA*F3/pi*real(Lmat(1,1)*Qmat(1,1)/(z1-zo1)*(mu_k
    (1)*ETA1-ETA2)+Lmat(1,2)*Qmat(2,1)/(z2-zo2)*(mu_k(2)*ETA1-ETA2));
%     HW(1,6)=HW(1,6)+OME(i)*XJA*F3/pi*real(Lmat(2,1)*Qmat(1,1)/(z1-zo1)*(mu_k
    (1)*ETA1-ETA2)+Lmat(2,2)*Qmat(2,1)/(z2-zo2)*(mu_k(2)*ETA1-ETA2));
%     HW(2,5)=HW(2,5)+OME(i)*XJA*F3/pi*real(Lmat(1,1)*Qmat(1,2)/(z1-zo1)*(mu_k
    (1)*ETA1-ETA2)+Lmat(1,2)*Qmat(2,2)/(z2-zo2)*(mu_k(2)*ETA1-ETA2));
%     HW(2,6)=HW(2,6)+OME(i)*XJA*F3/pi*real(Lmat(2,1)*Qmat(1,2)/(z1-zo1)*(mu_k
    (1)*ETA1-ETA2)+Lmat(2,2)*Qmat(2,2)/(z2-zo2)*(mu_k(2)*ETA1-ETA2));

   GW(1,1)=GW(1,1)-OME(i)*XJA*F1/pi*real(Amat(1,1)*Qmat(1,1)*log(z1-zo1)+Amat
       (1,2)*Qmat(2,1)*log(z2-zo2)+Amat(1,3)*Qmat(3,1)*log(z3-zo3));
   GW(1,2)=GW(1,2)-OME(i)*XJA*F1/pi*real(Amat(2,1)*Qmat(1,1)*log(z1-zo1)+Amat
       (2,2)*Qmat(2,1)*log(z2-zo2)+Amat(2,3)*Qmat(3,1)*log(z3-zo3));
   GW(1,3)=GW(1,3)-OME(i)*XJA*F1/pi*real(Amat(3,1)*Qmat(1,1)*log(z1-zo1)+Amat
       (3,2)*Qmat(2,1)*log(z2-zo2)+Amat(3,3)*Qmat(3,1)*log(z3-zo3));
   GW(1,4)=GW(1,4)-OME(i)*XJA*F2/pi*real(Amat(1,1)*Qmat(1,1)*log(z1-zo1)+Amat
       (1,2)*Qmat(2,1)*log(z2-zo2)+Amat(1,3)*Qmat(3,1)*log(z3-zo3));
   GW(1,5)=GW(1,5)-OME(i)*XJA*F2/pi*real(Amat(2,1)*Qmat(1,1)*log(z1-zo1)+Amat
       (2,2)*Qmat(2,1)*log(z2-zo2)+Amat(2,3)*Qmat(3,1)*log(z3-zo3));
   GW(1,6)=GW(1,6)-OME(i)*XJA*F2/pi*real(Amat(3,1)*Qmat(1,1)*log(z1-zo1)+Amat
       (3,2)*Qmat(2,1)*log(z2-zo2)+Amat(3,3)*Qmat(3,1)*log(z3-zo3));
   GW(1,7)=GW(1,7)-OME(i)*XJA*F3/pi*real(Amat(1,1)*Qmat(1,1)*log(z1-zo1)+Amat
       (1,2)*Qmat(2,1)*log(z2-zo2)+Amat(1,3)*Qmat(3,1)*log(z3-zo3));
   GW(1,8)=GW(1,8)-OME(i)*XJA*F3/pi*real(Amat(2,1)*Qmat(1,1)*log(z1-zo1)+Amat
       (2,2)*Qmat(2,1)*log(z2-zo2)+Amat(2,3)*Qmat(3,1)*log(z3-zo3));
```

```
    GW(1,9)=GW(1,9)-OME(i)*XJA*F3/pi*real(Amat(3,1)*Qmat(1,1)*log(z1-zo1)+Amat
        (3,2)*Qmat(2,1)*log(z2-zo2)+Amat(3,3)*Qmat(3,1)*log(z3-zo3));
    GW(2,1)=GW(2,1)-OME(i)*XJA*F1/pi*real(Amat(1,1)*Qmat(1,2)*log(z1-zo1)+Amat
        (1,2)*Qmat(2,2)*log(z2-zo2)+Amat(1,3)*Qmat(3,2)*log(z3-zo3));
    GW(2,2)=GW(2,2)-OME(i)*XJA*F1/pi*real(Amat(2,1)*Qmat(1,2)*log(z1-zo1)+Amat
        (2,2)*Qmat(2,2)*log(z2-zo2)+Amat(2,3)*Qmat(3,2)*log(z3-zo3));
    GW(2,3)=GW(2,3)-OME(i)*XJA*F1/pi*real(Amat(3,1)*Qmat(1,2)*log(z1-zo1)+Amat
        (3,2)*Qmat(2,2)*log(z2-zo2)+Amat(3,3)*Qmat(3,2)*log(z3-zo3));
    GW(2,4)=GW(2,4)-OME(i)*XJA*F2/pi*real(Amat(1,1)*Qmat(1,2)*log(z1-zo1)+Amat
        (1,2)*Qmat(2,2)*log(z2-zo2)+Amat(1,3)*Qmat(3,2)*log(z3-zo3));
    GW(2,5)=GW(2,5)-OME(i)*XJA*F2/pi*real(Amat(2,1)*Qmat(1,2)*log(z1-zo1)+Amat
        (2,2)*Qmat(2,2)*log(z2-zo2)+Amat(2,3)*Qmat(3,2)*log(z3-zo3));
    GW(2,6)=GW(2,6)-OME(i)*XJA*F2/pi*real(Amat(3,1)*Qmat(1,2)*log(z1-zo1)+Amat
        (3,2)*Qmat(2,2)*log(z2-zo2)+Amat(3,3)*Qmat(3,2)*log(z3-zo3));
    GW(2,7)=GW(2,7)-OME(i)*XJA*F3/pi*real(Amat(1,1)*Qmat(1,2)*log(z1-zo1)+Amat
        (1,2)*Qmat(2,2)*log(z2-zo2)+Amat(1,3)*Qmat(3,2)*log(z3-zo3));
    GW(2,8)=GW(2,8)-OME(i)*XJA*F3/pi*real(Amat(2,1)*Qmat(1,2)*log(z1-zo1)+Amat
        (2,2)*Qmat(2,2)*log(z2-zo2)+Amat(2,3)*Qmat(3,2)*log(z3-zo3));
    GW(2,9)=GW(2,9)-OME(i)*XJA*F3/pi*real(Amat(3,1)*Qmat(1,2)*log(z1-zo1)+Amat
        (3,2)*Qmat(2,2)*log(z2-zo2)+Amat(3,3)*Qmat(3,2)*log(z3-zo3));
    GW(3,1)=GW(3,1)-OME(i)*XJA*F1/pi*real(Amat(1,1)*Qmat(1,3)*log(z1-zo1)+Amat
        (1,2)*Qmat(2,3)*log(z2-zo2)+Amat(1,3)*Qmat(3,3)*log(z3-zo3));
    GW(3,2)=GW(3,2)-OME(i)*XJA*F1/pi*real(Amat(2,1)*Qmat(1,3)*log(z1-zo1)+Amat
        (2,2)*Qmat(2,3)*log(z2-zo2)+Amat(2,3)*Qmat(3,3)*log(z3-zo3));
    GW(3,3)=GW(3,3)-OME(i)*XJA*F1/pi*real(Amat(3,1)*Qmat(1,3)*log(z1-zo1)+Amat
        (3,2)*Qmat(2,3)*log(z2-zo2)+Amat(3,3)*Qmat(3,3)*log(z3-zo3));
    GW(3,4)=GW(3,4)-OME(i)*XJA*F2/pi*real(Amat(1,1)*Qmat(1,3)*log(z1-zo1)+Amat
        (1,2)*Qmat(2,3)*log(z2-zo2)+Amat(1,3)*Qmat(3,3)*log(z3-zo3));
    GW(3,5)=GW(3,5)-OME(i)*XJA*F2/pi*real(Amat(2,1)*Qmat(1,3)*log(z1-zo1)+Amat
        (2,2)*Qmat(2,3)*log(z2-zo2)+Amat(2,3)*Qmat(3,3)*log(z3-zo3));
    GW(3,6)=GW(3,6)-OME(i)*XJA*F2/pi*real(Amat(3,1)*Qmat(1,3)*log(z1-zo1)+Amat
        (3,2)*Qmat(2,3)*log(z2-zo2)+Amat(3,3)*Qmat(3,3)*log(z3-zo3));
    GW(3,7)=GW(3,7)-OME(i)*XJA*F3/pi*real(Amat(1,1)*Qmat(1,3)*log(z1-zo1)+Amat
        (1,2)*Qmat(2,3)*log(z2-zo2)+Amat(1,3)*Qmat(3,3)*log(z3-zo3));
    GW(3,8)=GW(3,8)-OME(i)*XJA*F3/pi*real(Amat(2,1)*Qmat(1,3)*log(z1-zo1)+Amat
        (2,2)*Qmat(2,3)*log(z2-zo2)+Amat(2,3)*Qmat(3,3)*log(z3-zo3));
    GW(3,9)=GW(3,9)-OME(i)*XJA*F3/pi*real(Amat(3,1)*Qmat(1,3)*log(z1-zo1)+Amat
        (3,2)*Qmat(2,3)*log(z2-zo2)+Amat(3,3)*Qmat(3,3)*log(z3-zo3));

    HW(1,1)=HW(1,1)+OME(i)*XJA*F1/pi*real(Lmat(1,1)*Qmat(1,1)/(z1-zo1)*(mu_k(1)
        *ETA1-ETA2)+Lmat(1,2)*Qmat(2,1)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat(1,3)*
        Qmat(3,1)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(1,2)=HW(1,2)+OME(i)*XJA*F1/pi*real(Lmat(2,1)*Qmat(1,1)/(z1-zo1)*(mu_k(1)
        *ETA1-ETA2)+Lmat(2,2)*Qmat(2,1)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat(2,3)*
        Qmat(3,1)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(1,3)=HW(1,3)+OME(i)*XJA*F1/pi*real(Lmat(3,1)*Qmat(1,1)/(z1-zo1)*(mu_k(1)
        *ETA1-ETA2)+Lmat(3,2)*Qmat(2,1)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat(3,3)*
        Qmat(3,1)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(1,4)=HW(1,4)+OME(i)*XJA*F2/pi*real(Lmat(1,1)*Qmat(1,1)/(z1-zo1)*(mu_k(1)
        *ETA1-ETA2)+Lmat(1,2)*Qmat(2,1)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat(1,3)*
        Qmat(3,1)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(1,5)=HW(1,5)+OME(i)*XJA*F2/pi*real(Lmat(2,1)*Qmat(1,1)/(z1-zo1)*(mu_k(1)
        *ETA1-ETA2)+Lmat(2,2)*Qmat(2,1)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat(2,3)*
        Qmat(3,1)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(1,6)=HW(1,6)+OME(i)*XJA*F2/pi*real(Lmat(3,1)*Qmat(1,1)/(z1-zo1)*(mu_k(1)
        *ETA1-ETA2)+Lmat(3,2)*Qmat(2,1)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat(3,3)*
        Qmat(3,1)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
```

```
    HW(1,7)=HW(1,7)+OME(i)*XJA*F3/pi*real(Lmat(1,1)*Qmat(1,1)/(z1-zo1)*(mu_k(1)
        *ETA1-ETA2)+Lmat(1,2)*Qmat(2,1)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat(1,3)*
        Qmat(3,1)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(1,8)=HW(1,8)+OME(i)*XJA*F3/pi*real(Lmat(2,1)*Qmat(1,1)/(z1-zo1)*(mu_k(1)
        *ETA1-ETA2)+Lmat(2,2)*Qmat(2,1)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat(2,3)*
        Qmat(3,1)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(1,9)=HW(1,9)+OME(i)*XJA*F3/pi*real(Lmat(3,1)*Qmat(1,1)/(z1-zo1)*(mu_k(1)
        *ETA1-ETA2)+Lmat(3,2)*Qmat(2,1)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat(3,3)*
        Qmat(3,1)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(2,1)=HW(2,1)+OME(i)*XJA*F1/pi*real(Lmat(1,1)*Qmat(1,2)/(z1-zo1)*(mu_k(1)
        *ETA1-ETA2)+Lmat(1,2)*Qmat(2,2)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat(1,3)*
        Qmat(3,2)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(2,2)=HW(2,2)+OME(i)*XJA*F1/pi*real(Lmat(2,1)*Qmat(1,2)/(z1-zo1)*(mu_k(1)
        *ETA1-ETA2)+Lmat(2,2)*Qmat(2,2)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat(2,3)*
        Qmat(3,2)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(2,3)=HW(2,3)+OME(i)*XJA*F1/pi*real(Lmat(3,1)*Qmat(1,2)/(z1-zo1)*(mu_k(1)
        *ETA1-ETA2)+Lmat(3,2)*Qmat(2,2)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat(3,3)*
        Qmat(3,2)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(2,4)=HW(2,4)+OME(i)*XJA*F2/pi*real(Lmat(1,1)*Qmat(1,2)/(z1-zo1)*(mu_k(1)
        *ETA1-ETA2)+Lmat(1,2)*Qmat(2,2)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat(1,3)*
        Qmat(3,2)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(2,5)=HW(2,5)+OME(i)*XJA*F2/pi*real(Lmat(2,1)*Qmat(1,2)/(z1-zo1)*(mu_k(1)
        *ETA1-ETA2)+Lmat(2,2)*Qmat(2,2)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat(2,3)*
        Qmat(3,2)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(2,6)=HW(2,6)+OME(i)*XJA*F2/pi*real(Lmat(3,1)*Qmat(1,2)/(z1-zo1)*(mu_k(1)
        *ETA1-ETA2)+Lmat(3,2)*Qmat(2,2)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat(3,3)*
        Qmat(3,2)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(2,7)=HW(2,7)+OME(i)*XJA*F3/pi*real(Lmat(1,1)*Qmat(1,2)/(z1-zo1)*(mu_k(1)
        *ETA1-ETA2)+Lmat(1,2)*Qmat(2,2)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat(1,3)*
        Qmat(3,2)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(2,8)=HW(2,8)+OME(i)*XJA*F3/pi*real(Lmat(2,1)*Qmat(1,2)/(z1-zo1)*(mu_k(1)
        *ETA1-ETA2)+Lmat(2,2)*Qmat(2,2)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat(2,3)*
        Qmat(3,2)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(2,9)=HW(2,9)+OME(i)*XJA*F3/pi*real(Lmat(3,1)*Qmat(1,2)/(z1-zo1)*(mu_k(1)
        *ETA1-ETA2)+Lmat(3,2)*Qmat(2,2)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat(3,3)*
        Qmat(3,2)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(3,1)=HW(3,1)+OME(i)*XJA*F1/pi*real(Lmat(1,1)*Qmat(1,3)/(z1-zo1)*(mu_k(1)
        *ETA1-ETA2)+Lmat(1,2)*Qmat(2,3)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat(1,3)*
        Qmat(3,3)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(3,2)=HW(3,2)+OME(i)*XJA*F1/pi*real(Lmat(2,1)*Qmat(1,3)/(z1-zo1)*(mu_k(1)
        *ETA1-ETA2)+Lmat(2,2)*Qmat(2,3)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat(2,3)*
        Qmat(3,3)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(3,3)=HW(3,3)+OME(i)*XJA*F1/pi*real(Lmat(3,1)*Qmat(1,3)/(z1-zo1)*(mu_k(1)
        *ETA1-ETA2)+Lmat(3,2)*Qmat(2,3)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat(3,3)*
        Qmat(3,3)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(3,4)=HW(3,4)+OME(i)*XJA*F2/pi*real(Lmat(1,1)*Qmat(1,3)/(z1-zo1)*(mu_k(1)
        *ETA1-ETA2)+Lmat(1,2)*Qmat(2,3)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat(1,3)*
        Qmat(3,3)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(3,5)=HW(3,5)+OME(i)*XJA*F2/pi*real(Lmat(2,1)*Qmat(1,3)/(z1-zo1)*(mu_k(1)
        *ETA1-ETA2)+Lmat(2,2)*Qmat(2,3)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat(2,3)*
        Qmat(3,3)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(3,6)=HW(3,6)+OME(i)*XJA*F2/pi*real(Lmat(3,1)*Qmat(1,3)/(z1-zo1)*(mu_k(1)
        *ETA1-ETA2)+Lmat(3,2)*Qmat(2,3)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat(3,3)*
        Qmat(3,3)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(3,7)=HW(3,7)+OME(i)*XJA*F3/pi*real(Lmat(1,1)*Qmat(1,3)/(z1-zo1)*(mu_k(1)
        *ETA1-ETA2)+Lmat(1,2)*Qmat(2,3)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat(1,3)*
        Qmat(3,3)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
```

```
    HW(3,8)=HW(3,8)+OME(i)*XJA*F3/pi*real(Lmat(2,1)*Qmat(1,3)/(z1-zo1)*(mu_k(1)
        *ETA1-ETA2)+Lmat(2,2)*Qmat(2,3)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat(2,3)*
        Qmat(3,3)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(3,9)=HW(3,9)+OME(i)*XJA*F3/pi*real(Lmat(3,1)*Qmat(1,3)/(z1-zo1)*(mu_k(1)
        *ETA1-ETA2)+Lmat(3,2)*Qmat(2,3)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat(3,3)*
        Qmat(3,3)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
end

 GW=GW/JAC;
 HW=HW/JAC;

end
```

B.7 SSTANI

```
function [HW]=SSTANI(XP,YP,X1,Y1,X2,Y2,X3,Y3,inicio,final,COL,knotVec,weight,
    active,Qmat,Lmat,mu_k,Xfirst,Yfirst,vecpos)

if COL<inicio
    COL=final;
end

global NURBS dNURBS

GI=[0.9739065285,-0.9739065285,0.8650633665,-0.8650633665,0.6794095683,
-0.6794095683,0.4333953941,-0.4333953941,0.1488743389,-0.1488743389];
OME=[0.0666713443,0.0666713443,0.1494513491,0.1494513491,0.2190863625,
0.2190863625,0.2692667193,0.2692667193,0.2955242247,0.2955242247];
HW=zeros(3,9);
JAC=2/(final-inicio);
NODO=2;
if COL~=inicio && COL~=final
    NODO=2;
elseif COL==inicio
     NODO=1;
elseif COL==final
    NODO=3;
end

% Transformation to complex plane

zo1=XP+YP*mu_k(1);
zo2=XP+YP*mu_k(2);
zo3=XP+YP*mu_k(3);

if NODO==1 || NODO==3

for i=1:10

   % Compute the values of the shape functions at the integration points.

    if XP==Xfirst && YP==Yfirst
    [F1,dR1]=NURBSbasis_matlab(active,2,(GI(i)*(final-inicio)+final+inicio)/2,
        knotVec,weight);
```

```
[F2,dR2]=NURBSbasis_matlab(active+1,2,(GI(i)*(final-inicio)+final+inicio)
    /2,knotVec,weight);
[F3,dR3]=NURBSbasis_matlab(active+2,2,(GI(i)*(final-inicio)+final+inicio)
    /2,knotVec,weight);
NURBS(active,10*vecpos-10+i)=F1;
NURBS(active+1,10*vecpos-10+i)=F2;
NURBS(active+2,10*vecpos-10+i)=F3;
dNURBS(active,10*vecpos-10+i)=dR1;
dNURBS(active+1,10*vecpos-10+i)=dR2;
dNURBS(active+2,10*vecpos-10+i)=dR3;
else
F1=NURBS(active,10*vecpos-10+i);
F2=NURBS(active+1,10*vecpos-10+i);
F3=NURBS(active+2,10*vecpos-10+i);
dR1=dNURBS(active,10*vecpos-10+i);
dR2=dNURBS(active+1,10*vecpos-10+i);
dR3=dNURBS(active+2,10*vecpos-10+i);
end

% Compute geometrical properties at the integration points.

XCO=X1*F1+X2*F2+X3*F3;
YCO=Y1*F1+Y2*F2+Y3*F3;
A=dR1*X1+dR2*X2+dR3*X3;
B=dR1*Y1+dR2*Y2+dR3*Y3;
XJA=sqrt(A^2+B^2);
ETA1=B/XJA;
ETA2=-A/XJA;

% Transformation to complex plane

z1=XCO+YCO*mu_k(1);
z2=XCO+YCO*mu_k(2);
z3=XCO+YCO*mu_k(3);

% Compute GW and HW matrices.

HW(1,1)=HW(1,1)+OME(i)*XJA/JAC*F1/pi*real(Lmat(1,1)*Qmat(1,1)/(z1-zo1)*(mu_
    k(1)*ETA1-ETA2)+Lmat(1,2)*Qmat(2,1)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
    (1,3)*Qmat(3,1)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
HW(1,2)=HW(1,2)+OME(i)*XJA/JAC*F1/pi*real(Lmat(2,1)*Qmat(1,1)/(z1-zo1)*(mu_
    k(1)*ETA1-ETA2)+Lmat(2,2)*Qmat(2,1)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
    (2,3)*Qmat(3,1)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
HW(1,3)=HW(1,3)+OME(i)*XJA/JAC*F1/pi*real(Lmat(3,1)*Qmat(1,1)/(z1-zo1)*(mu_
    k(1)*ETA1-ETA2)+Lmat(3,2)*Qmat(2,1)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
    (3,3)*Qmat(3,1)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
HW(1,4)=HW(1,4)+OME(i)*XJA/JAC*F2/pi*real(Lmat(1,1)*Qmat(1,1)/(z1-zo1)*(mu_
    k(1)*ETA1-ETA2)+Lmat(1,2)*Qmat(2,1)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
    (1,3)*Qmat(3,1)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
HW(1,5)=HW(1,5)+OME(i)*XJA/JAC*F2/pi*real(Lmat(2,1)*Qmat(1,1)/(z1-zo1)*(mu_
    k(1)*ETA1-ETA2)+Lmat(2,2)*Qmat(2,1)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
    (2,3)*Qmat(3,1)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
HW(1,6)=HW(1,6)+OME(i)*XJA/JAC*F2/pi*real(Lmat(3,1)*Qmat(1,1)/(z1-zo1)*(mu_
    k(1)*ETA1-ETA2)+Lmat(3,2)*Qmat(2,1)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
    (3,3)*Qmat(3,1)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
```

```
    HW(1,7)=HW(1,7)+OME(i)*XJA/JAC*F3/pi*real(Lmat(1,1)*Qmat(1,1)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(1,2)*Qmat(2,1)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (1,3)*Qmat(3,1)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(1,8)=HW(1,8)+OME(i)*XJA/JAC*F3/pi*real(Lmat(2,1)*Qmat(1,1)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(2,2)*Qmat(2,1)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (2,3)*Qmat(3,1)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(1,9)=HW(1,9)+OME(i)*XJA/JAC*F3/pi*real(Lmat(3,1)*Qmat(1,1)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(3,2)*Qmat(2,1)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (3,3)*Qmat(3,1)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(2,1)=HW(2,1)+OME(i)*XJA/JAC*F1/pi*real(Lmat(1,1)*Qmat(1,2)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(1,2)*Qmat(2,2)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (1,3)*Qmat(3,2)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(2,2)=HW(2,2)+OME(i)*XJA/JAC*F1/pi*real(Lmat(2,1)*Qmat(1,2)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(2,2)*Qmat(2,2)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (2,3)*Qmat(3,2)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(2,3)=HW(2,3)+OME(i)*XJA/JAC*F1/pi*real(Lmat(3,1)*Qmat(1,2)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(3,2)*Qmat(2,2)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (3,3)*Qmat(3,2)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(2,4)=HW(2,4)+OME(i)*XJA/JAC*F2/pi*real(Lmat(1,1)*Qmat(1,2)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(1,2)*Qmat(2,2)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (1,3)*Qmat(3,2)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(2,5)=HW(2,5)+OME(i)*XJA/JAC*F2/pi*real(Lmat(2,1)*Qmat(1,2)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(2,2)*Qmat(2,2)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (2,3)*Qmat(3,2)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(2,6)=HW(2,6)+OME(i)*XJA/JAC*F2/pi*real(Lmat(3,1)*Qmat(1,2)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(3,2)*Qmat(2,2)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (3,3)*Qmat(3,2)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(2,7)=HW(2,7)+OME(i)*XJA/JAC*F3/pi*real(Lmat(1,1)*Qmat(1,2)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(1,2)*Qmat(2,2)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (1,3)*Qmat(3,2)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(2,8)=HW(2,8)+OME(i)*XJA/JAC*F3/pi*real(Lmat(2,1)*Qmat(1,2)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(2,2)*Qmat(2,2)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (2,3)*Qmat(3,2)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(2,9)=HW(2,9)+OME(i)*XJA/JAC*F3/pi*real(Lmat(3,1)*Qmat(1,2)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(3,2)*Qmat(2,2)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (3,3)*Qmat(3,2)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(3,1)=HW(3,1)+OME(i)*XJA/JAC*F1/pi*real(Lmat(1,1)*Qmat(1,3)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(1,2)*Qmat(2,3)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (1,3)*Qmat(3,3)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(3,2)=HW(3,2)+OME(i)*XJA/JAC*F1/pi*real(Lmat(2,1)*Qmat(1,3)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(2,2)*Qmat(2,3)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (2,3)*Qmat(3,3)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(3,3)=HW(3,3)+OME(i)*XJA/JAC*F1/pi*real(Lmat(3,1)*Qmat(1,3)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(3,2)*Qmat(2,3)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (3,3)*Qmat(3,3)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(3,4)=HW(3,4)+OME(i)*XJA/JAC*F2/pi*real(Lmat(1,1)*Qmat(1,3)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(1,2)*Qmat(2,3)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (1,3)*Qmat(3,3)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(3,5)=HW(3,5)+OME(i)*XJA/JAC*F2/pi*real(Lmat(2,1)*Qmat(1,3)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(2,2)*Qmat(2,3)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (2,3)*Qmat(3,3)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(3,6)=HW(3,6)+OME(i)*XJA/JAC*F2/pi*real(Lmat(3,1)*Qmat(1,3)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(3,2)*Qmat(2,3)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (3,3)*Qmat(3,3)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(3,7)=HW(3,7)+OME(i)*XJA/JAC*F3/pi*real(Lmat(1,1)*Qmat(1,3)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(1,2)*Qmat(2,3)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (1,3)*Qmat(3,3)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
```

```
    HW(3,8)=HW(3,8)+OME(i)*XJA/JAC*F3/pi*real(Lmat(2,1)*Qmat(1,3)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(2,2)*Qmat(2,3)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (2,3)*Qmat(3,3)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    HW(3,9)=HW(3,9)+OME(i)*XJA/JAC*F3/pi*real(Lmat(3,1)*Qmat(1,3)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(3,2)*Qmat(2,3)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (3,3)*Qmat(3,3)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
end

if NODO==1

    [F1,dR11]=NURBSbasis_matlab(active,2,inicio+(final-inicio)*0.0001,knotVec,
        weight);
    [F2,dR12]=NURBSbasis_matlab(active+1,2,inicio+(final-inicio)*0.0001,knotVec
        ,weight);
    [F3,dR13]=NURBSbasis_matlab(active+2,2,inicio+(final-inicio)*0.0001,knotVec
        ,weight);
    A=dR11*X1+dR12*X2+dR13*X3;
    B=dR11*Y1+dR12*Y2+dR13*Y3;
    XJA=sqrt(A^2+B^2);

    XCO=X1*F1+X2*F2+X3*F3;
    YCO=Y1*F1+Y2*F2+Y3*F3;
    z1=XCO+YCO*mu_k(1);
    z2=XCO+YCO*mu_k(2);
    z3=XCO+YCO*mu_k(3);
    ETA1=B/XJA;
    ETA2=-A/XJA;
    CONST11=XJA*(final-inicio)*0.0001/pi*real(Lmat(1,1)*Qmat(1,1)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(1,2)*Qmat(2,1)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (1,3)*Qmat(3,1)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    CONST12=XJA*(final-inicio)*0.0001/pi*real(Lmat(2,1)*Qmat(1,1)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(2,2)*Qmat(2,1)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (2,3)*Qmat(3,1)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    CONST13=XJA*(final-inicio)*0.0001/pi*real(Lmat(3,1)*Qmat(1,1)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(3,2)*Qmat(2,1)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (3,3)*Qmat(3,1)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    CONST21=XJA*(final-inicio)*0.0001/pi*real(Lmat(1,1)*Qmat(1,2)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(1,2)*Qmat(2,2)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (1,3)*Qmat(3,2)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    CONST22=XJA*(final-inicio)*0.0001/pi*real(Lmat(2,1)*Qmat(1,2)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(2,2)*Qmat(2,2)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (2,3)*Qmat(3,2)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    CONST23=XJA*(final-inicio)*0.0001/pi*real(Lmat(3,1)*Qmat(1,2)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(3,2)*Qmat(2,2)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (3,3)*Qmat(3,2)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    CONST31=XJA*(final-inicio)*0.0001/pi*real(Lmat(1,1)*Qmat(1,3)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(1,2)*Qmat(2,3)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (1,3)*Qmat(3,3)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    CONST32=XJA*(final-inicio)*0.0001/pi*real(Lmat(2,1)*Qmat(1,3)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(2,2)*Qmat(2,3)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (2,3)*Qmat(3,3)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    CONST33=XJA*(final-inicio)*0.0001/pi*real(Lmat(3,1)*Qmat(1,3)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(3,2)*Qmat(2,3)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (3,3)*Qmat(3,3)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));

end
```

```
if NODO==3

    [F1,dR31]=NURBSbasis_matlab(active,2,final-(final-inicio)*0.0001,knotVec,
        weight);
    [F2,dR32]=NURBSbasis_matlab(active+1,2,final-(final-inicio)*0.0001,knotVec,
        weight);
    [F3,dR33]=NURBSbasis_matlab(active+2,2,final-(final-inicio)*0.0001,knotVec,
        weight);
    A=dR31*X1+dR32*X2+dR33*X3;
    B=dR31*Y1+dR32*Y2+dR33*Y3;
    XJA=sqrt(A^2+B^2);

    XCO=X1*F1+X2*F2+X3*F3;
    YCO=Y1*F1+Y2*F2+Y3*F3;
    z1=XCO+YCO*mu_k(1);
    z2=XCO+YCO*mu_k(2);
    ETA1=B/XJA;
    ETA2=-A/XJA;
    CONST11=XJA*(final-inicio)*0.0001/pi*real(Lmat(1,1)*Qmat(1,1)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(1,2)*Qmat(2,1)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (1,3)*Qmat(3,1)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    CONST12=XJA*(final-inicio)*0.0001/pi*real(Lmat(2,1)*Qmat(1,1)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(2,2)*Qmat(2,1)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (2,3)*Qmat(3,1)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    CONST13=XJA*(final-inicio)*0.0001/pi*real(Lmat(3,1)*Qmat(1,1)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(3,2)*Qmat(2,1)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (3,3)*Qmat(3,1)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    CONST21=XJA*(final-inicio)*0.0001/pi*real(Lmat(1,1)*Qmat(1,2)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(1,2)*Qmat(2,2)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (1,3)*Qmat(3,2)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    CONST22=XJA*(final-inicio)*0.0001/pi*real(Lmat(2,1)*Qmat(1,2)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(2,2)*Qmat(2,2)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (2,3)*Qmat(3,2)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    CONST23=XJA*(final-inicio)*0.0001/pi*real(Lmat(3,1)*Qmat(1,2)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(3,2)*Qmat(2,2)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (3,3)*Qmat(3,2)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    CONST31=XJA*(final-inicio)*0.0001/pi*real(Lmat(1,1)*Qmat(1,3)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(1,2)*Qmat(2,3)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (1,3)*Qmat(3,3)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    CONST32=XJA*(final-inicio)*0.0001/pi*real(Lmat(2,1)*Qmat(1,3)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(2,2)*Qmat(2,3)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (2,3)*Qmat(3,3)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));
    CONST33=XJA*(final-inicio)*0.0001/pi*real(Lmat(3,1)*Qmat(1,3)/(z1-zo1)*(mu_
        k(1)*ETA1-ETA2)+Lmat(3,2)*Qmat(2,3)/(z2-zo2)*(mu_k(2)*ETA1-ETA2)+Lmat
        (3,3)*Qmat(3,3)/(z3-zo3)*(mu_k(3)*ETA1-ETA2));;

end

    HW(1,1)=HW(1,1)+F1*log(abs(XJA/JAC))*CONST11;
    HW(1,2)=HW(1,2)+F1*log(abs(XJA/JAC))*CONST12;
    HW(1,3)=HW(1,3)+F1*log(abs(XJA/JAC))*CONST13;
    HW(1,4)=HW(1,4)+F2*log(abs(XJA/JAC))*CONST11;
    HW(1,5)=HW(1,5)+F2*log(abs(XJA/JAC))*CONST12;
    HW(1,6)=HW(1,6)+F2*log(abs(XJA/JAC))*CONST13;
    HW(1,7)=HW(1,7)+F3*log(abs(XJA/JAC))*CONST11;
    HW(1,8)=HW(1,8)+F3*log(abs(XJA/JAC))*CONST12;
    HW(1,9)=HW(1,9)+F3*log(abs(XJA/JAC))*CONST13;
```

```
    HW(2,1)=HW(2,1)+F1*log(abs(XJA/JAC))*CONST21;
    HW(2,2)=HW(2,2)+F1*log(abs(XJA/JAC))*CONST22;
    HW(2,3)=HW(2,3)+F1*log(abs(XJA/JAC))*CONST23;
    HW(2,4)=HW(2,4)+F2*log(abs(XJA/JAC))*CONST21;
    HW(2,5)=HW(2,5)+F2*log(abs(XJA/JAC))*CONST22;
    HW(2,6)=HW(2,6)+F2*log(abs(XJA/JAC))*CONST23;
    HW(2,7)=HW(2,7)+F3*log(abs(XJA/JAC))*CONST21;
    HW(2,8)=HW(2,8)+F3*log(abs(XJA/JAC))*CONST22;
    HW(2,9)=HW(2,9)+F3*log(abs(XJA/JAC))*CONST23;
    HW(3,1)=HW(3,1)+F1*log(abs(XJA/JAC))*CONST31;
    HW(3,2)=HW(3,2)+F1*log(abs(XJA/JAC))*CONST32;
    HW(3,3)=HW(3,3)+F1*log(abs(XJA/JAC))*CONST33;
    HW(3,4)=HW(3,4)+F2*log(abs(XJA/JAC))*CONST31;
    HW(3,5)=HW(3,5)+F2*log(abs(XJA/JAC))*CONST32;
    HW(3,6)=HW(3,6)+F2*log(abs(XJA/JAC))*CONST33;
    HW(3,7)=HW(3,7)+F3*log(abs(XJA/JAC))*CONST31;
    HW(3,8)=HW(3,8)+F3*log(abs(XJA/JAC))*CONST32;
    HW(3,9)=HW(3,9)+F3*log(abs(XJA/JAC))*CONST33;
end

if NODO==2

    for i=1:10

    [F11,dR11]=NURBSbasis_matlab(active,2,(GI(i)*(COL-inicio)+COL+inicio)/2,
        knotVec,weight);
    [F21,dR21]=NURBSbasis_matlab(active+1,2,(GI(i)*(COL-inicio)+COL+inicio)/2,
        knotVec,weight);
    [F31,dR31]=NURBSbasis_matlab(active+2,2,(GI(i)*(COL-inicio)+COL+inicio)/2,
        knotVec,weight);
    [F12,dR12]=NURBSbasis_matlab(active,2,(GI(i)*(final-COL)+COL+final)/2,
        knotVec,weight);
    [F22,dR22]=NURBSbasis_matlab(active+1,2,(GI(i)*(final-COL)+COL+final)/2,
        knotVec,weight);
    [F32,dR32]=NURBSbasis_matlab(active+2,2,(GI(i)*(final-COL)+COL+final)/2,
        knotVec,weight);

    XCO1=X1*F11+X2*F21+X3*F31;
    YCO1=Y1*F11+Y2*F21+Y3*F31;
    A1=dR11*X1+dR21*X2+dR31*X3;
    B1=dR11*Y1+dR21*Y2+dR31*Y3;
    XJA1=sqrt(A1^2+B1^2);
    ETA11=B1/XJA1;
    ETA12=-A1/XJA1;

    XCO2=X1*F12+X2*F22+X3*F32;
    YCO2=Y1*F12+Y2*F22+Y3*F32;
    A2=dR12*X1+dR22*X2+dR32*X3;
    B2=dR12*Y1+dR22*Y2+dR32*Y3;
    XJA2=sqrt(A2^2+B2^2);
    ETA21=B2/XJA2;
    ETA22=-A2/XJA2;

    z11=XCO1+YCO1*mu_k(1);
    z12=XCO1+YCO1*mu_k(2);
    z13=XCO1+YCO1*mu_k(3);
    z21=XCO2+YCO2*mu_k(1);
```

```
    z22=XCO2+YCO2*mu_k(2);
    z23=XCO2+YCO2*mu_k(3);

    HW(1,1)=HW(1,1)+OME(i)*(COL-inicio)/2*XJA1*F11/pi*real(Lmat(1,1)*Qmat(1,1)
        /(z11-zo1)*(mu_k(1)*ETA11-ETA12)+Lmat(1,2)*Qmat(2,1)/(z12-zo2)*(mu_k(2)
        *ETA11-ETA12)+Lmat(1,3)*Qmat(3,1)/(z13-zo3)*(mu_k(3)*ETA11-ETA12))+OME(
        i)*(final-COL)/2*XJA2*F12/pi*real(Lmat(1,1)*Qmat(1,1)/(z21-zo1)*(mu_k
        (1)*ETA21-ETA22)+Lmat(1,2)*Qmat(2,1)/(z22-zo2)*(mu_k(2)*ETA21-ETA22)+
        Lmat(1,3)*Qmat(3,1)/(z23-zo3)*(mu_k(3)*ETA21-ETA22));
    HW(1,2)=HW(1,2)+OME(i)*(COL-inicio)/2*XJA1*F11/pi*real(Lmat(2,1)*Qmat(1,1)
        /(z11-zo1)*(mu_k(1)*ETA11-ETA12)+Lmat(2,2)*Qmat(2,1)/(z12-zo2)*(mu_k(2)
        *ETA11-ETA12)+Lmat(2,3)*Qmat(3,1)/(z13-zo3)*(mu_k(3)*ETA11-ETA12))+OME(
        i)*(final-COL)/2*XJA2*F12/pi*real(Lmat(2,1)*Qmat(1,1)/(z21-zo1)*(mu_k
        (1)*ETA21-ETA22)+Lmat(2,2)*Qmat(2,1)/(z22-zo2)*(mu_k(2)*ETA21-ETA22)+
        Lmat(2,3)*Qmat(3,1)/(z23-zo3)*(mu_k(3)*ETA21-ETA22));
    HW(1,3)=HW(1,3)+OME(i)*(COL-inicio)/2*XJA1*F11/pi*real(Lmat(3,1)*Qmat(1,1)
        /(z11-zo1)*(mu_k(1)*ETA11-ETA12)+Lmat(3,2)*Qmat(2,1)/(z12-zo2)*(mu_k(2)
        *ETA11-ETA12)+Lmat(3,3)*Qmat(3,1)/(z13-zo3)*(mu_k(3)*ETA11-ETA12))+OME(
        i)*(final-COL)/2*XJA2*F12/pi*real(Lmat(3,1)*Qmat(1,1)/(z21-zo1)*(mu_k
        (1)*ETA21-ETA22)+Lmat(3,2)*Qmat(2,1)/(z22-zo2)*(mu_k(2)*ETA21-ETA22)+
        Lmat(3,3)*Qmat(3,1)/(z23-zo3)*(mu_k(3)*ETA21-ETA22));
    HW(1,4)=HW(1,4)+OME(i)*(COL-inicio)/2*XJA1*F21/pi*real(Lmat(1,1)*Qmat(1,1)
        /(z11-zo1)*(mu_k(1)*ETA11-ETA12)+Lmat(1,2)*Qmat(2,1)/(z12-zo2)*(mu_k(2)
        *ETA11-ETA12)+Lmat(1,3)*Qmat(3,1)/(z13-zo3)*(mu_k(3)*ETA11-ETA12))+OME(
        i)*(final-COL)/2*XJA2*F22/pi*real(Lmat(1,1)*Qmat(1,1)/(z21-zo1)*(mu_k
        (1)*ETA21-ETA22)+Lmat(1,2)*Qmat(2,1)/(z22-zo2)*(mu_k(2)*ETA21-ETA22)+
        Lmat(1,3)*Qmat(3,1)/(z23-zo3)*(mu_k(3)*ETA21-ETA22));
    HW(1,5)=HW(1,5)+OME(i)*(COL-inicio)/2*XJA1*F21/pi*real(Lmat(2,1)*Qmat(1,1)
        /(z11-zo1)*(mu_k(1)*ETA11-ETA12)+Lmat(2,2)*Qmat(2,1)/(z12-zo2)*(mu_k(2)
        *ETA11-ETA12)+Lmat(2,3)*Qmat(3,1)/(z13-zo3)*(mu_k(3)*ETA11-ETA12))+OME(
        i)*(final-COL)/2*XJA2*F22/pi*real(Lmat(2,1)*Qmat(1,1)/(z21-zo1)*(mu_k
        (1)*ETA21-ETA22)+Lmat(2,2)*Qmat(2,1)/(z22-zo2)*(mu_k(2)*ETA21-ETA22)+
        Lmat(2,3)*Qmat(3,1)/(z23-zo3)*(mu_k(3)*ETA21-ETA22));
    HW(1,6)=HW(1,6)+OME(i)*(COL-inicio)/2*XJA1*F21/pi*real(Lmat(3,1)*Qmat(1,1)
        /(z11-zo1)*(mu_k(1)*ETA11-ETA12)+Lmat(3,2)*Qmat(2,1)/(z12-zo2)*(mu_k(2)
        *ETA11-ETA12)+Lmat(3,3)*Qmat(3,1)/(z13-zo3)*(mu_k(3)*ETA11-ETA12))+OME(
        i)*(final-COL)/2*XJA2*F22/pi*real(Lmat(3,1)*Qmat(1,1)/(z21-zo1)*(mu_k
        (1)*ETA21-ETA22)+Lmat(3,2)*Qmat(2,1)/(z22-zo2)*(mu_k(2)*ETA21-ETA22)+
        Lmat(3,3)*Qmat(3,1)/(z23-zo3)*(mu_k(3)*ETA21-ETA22));
    HW(1,7)=HW(1,7)+OME(i)*(COL-inicio)/2*XJA1*F31/pi*real(Lmat(1,1)*Qmat(1,1)
        /(z11-zo1)*(mu_k(1)*ETA11-ETA12)+Lmat(1,2)*Qmat(2,1)/(z12-zo2)*(mu_k(2)
        *ETA11-ETA12)+Lmat(1,3)*Qmat(3,1)/(z13-zo3)*(mu_k(3)*ETA11-ETA12))+OME(
        i)*(final-COL)/2*XJA2*F32/pi*real(Lmat(1,1)*Qmat(1,1)/(z21-zo1)*(mu_k
        (1)*ETA21-ETA22)+Lmat(1,2)*Qmat(2,1)/(z22-zo2)*(mu_k(2)*ETA21-ETA22)+
        Lmat(1,3)*Qmat(3,1)/(z23-zo3)*(mu_k(3)*ETA21-ETA22));
    HW(1,8)=HW(1,8)+OME(i)*(COL-inicio)/2*XJA1*F31/pi*real(Lmat(2,1)*Qmat(1,1)
        /(z11-zo1)*(mu_k(1)*ETA11-ETA12)+Lmat(2,2)*Qmat(2,1)/(z12-zo2)*(mu_k(2)
        *ETA11-ETA12)+Lmat(2,3)*Qmat(3,1)/(z13-zo3)*(mu_k(3)*ETA11-ETA12))+OME(
        i)*(final-COL)/2*XJA2*F32/pi*real(Lmat(2,1)*Qmat(1,1)/(z21-zo1)*(mu_k
        (1)*ETA21-ETA22)+Lmat(2,2)*Qmat(2,1)/(z22-zo2)*(mu_k(2)*ETA21-ETA22)+
        Lmat(2,3)*Qmat(3,1)/(z23-zo3)*(mu_k(3)*ETA21-ETA22));
    HW(1,9)=HW(1,9)+OME(i)*(COL-inicio)/2*XJA1*F31/pi*real(Lmat(3,1)*Qmat(1,1)
        /(z11-zo1)*(mu_k(1)*ETA11-ETA12)+Lmat(3,2)*Qmat(2,1)/(z12-zo2)*(mu_k(2)
        *ETA11-ETA12)+Lmat(3,3)*Qmat(3,1)/(z13-zo3)*(mu_k(3)*ETA11-ETA12))+OME(
        i)*(final-COL)/2*XJA2*F32/pi*real(Lmat(3,1)*Qmat(1,1)/(z21-zo1)*(mu_k
        (1)*ETA21-ETA22)+Lmat(3,2)*Qmat(2,1)/(z22-zo2)*(mu_k(2)*ETA21-ETA22)+
        Lmat(3,3)*Qmat(3,1)/(z23-zo3)*(mu_k(3)*ETA21-ETA22));
```

```
    HW(2,1)=HW(2,1)+OME(i)*(COL-inicio)/2*XJA1*F11/pi*real(Lmat(1,1)*Qmat(1,2)
        /(z11-zo1)*(mu_k(1)*ETA11-ETA12)+Lmat(1,2)*Qmat(2,2)/(z12-zo2)*(mu_k(2)
        *ETA11-ETA12)+Lmat(1,3)*Qmat(3,2)/(z13-zo3)*(mu_k(3)*ETA11-ETA12))+OME(
        i)*(final-COL)/2*XJA2*F12/pi*real(Lmat(1,1)*Qmat(1,2)/(z21-zo1)*(mu_k
        (1)*ETA21-ETA22)+Lmat(1,2)*Qmat(2,2)/(z22-zo2)*(mu_k(2)*ETA21-ETA22)+
        Lmat(1,3)*Qmat(3,2)/(z23-zo3)*(mu_k(3)*ETA21-ETA22));
    HW(2,2)=HW(2,2)+OME(i)*(COL-inicio)/2*XJA1*F11/pi*real(Lmat(2,1)*Qmat(1,2)
        /(z11-zo1)*(mu_k(1)*ETA11-ETA12)+Lmat(2,2)*Qmat(2,2)/(z12-zo2)*(mu_k(2)
        *ETA11-ETA12)+Lmat(2,3)*Qmat(3,2)/(z13-zo3)*(mu_k(3)*ETA11-ETA12))+OME(
        i)*(final-COL)/2*XJA2*F12/pi*real(Lmat(2,1)*Qmat(1,2)/(z21-zo1)*(mu_k
        (1)*ETA21-ETA22)+Lmat(2,2)*Qmat(2,2)/(z22-zo2)*(mu_k(2)*ETA21-ETA22)+
        Lmat(2,3)*Qmat(3,2)/(z23-zo3)*(mu_k(3)*ETA21-ETA22));
    HW(2,3)=HW(2,3)+OME(i)*(COL-inicio)/2*XJA1*F11/pi*real(Lmat(3,1)*Qmat(1,2)
        /(z11-zo1)*(mu_k(1)*ETA11-ETA12)+Lmat(3,2)*Qmat(2,2)/(z12-zo2)*(mu_k(2)
        *ETA11-ETA12)+Lmat(3,3)*Qmat(3,2)/(z13-zo3)*(mu_k(3)*ETA11-ETA12))+OME(
        i)*(final-COL)/2*XJA2*F12/pi*real(Lmat(3,1)*Qmat(1,2)/(z21-zo1)*(mu_k
        (1)*ETA21-ETA22)+Lmat(3,2)*Qmat(2,2)/(z22-zo2)*(mu_k(2)*ETA21-ETA22)+
        Lmat(3,3)*Qmat(3,2)/(z23-zo3)*(mu_k(3)*ETA21-ETA22));
    HW(2,4)=HW(2,4)+OME(i)*(COL-inicio)/2*XJA1*F21/pi*real(Lmat(1,1)*Qmat(1,2)
        /(z11-zo1)*(mu_k(1)*ETA11-ETA12)+Lmat(1,2)*Qmat(2,2)/(z12-zo2)*(mu_k(2)
        *ETA11-ETA12)+Lmat(1,3)*Qmat(3,2)/(z13-zo3)*(mu_k(3)*ETA11-ETA12))+OME(
        i)*(final-COL)/2*XJA2*F22/pi*real(Lmat(1,1)*Qmat(1,2)/(z21-zo1)*(mu_k
        (1)*ETA21-ETA22)+Lmat(1,2)*Qmat(2,2)/(z22-zo2)*(mu_k(2)*ETA21-ETA22)+
        Lmat(1,3)*Qmat(3,2)/(z23-zo3)*(mu_k(3)*ETA21-ETA22));
    HW(2,5)=HW(2,5)+OME(i)*(COL-inicio)/2*XJA1*F21/pi*real(Lmat(2,1)*Qmat(1,2)
        /(z11-zo1)*(mu_k(1)*ETA11-ETA12)+Lmat(2,2)*Qmat(2,2)/(z12-zo2)*(mu_k(2)
        *ETA11-ETA12)+Lmat(2,3)*Qmat(3,2)/(z13-zo3)*(mu_k(3)*ETA11-ETA12))+OME(
        i)*(final-COL)/2*XJA2*F22/pi*real(Lmat(2,1)*Qmat(1,2)/(z21-zo1)*(mu_k
        (1)*ETA21-ETA22)+Lmat(2,2)*Qmat(2,2)/(z22-zo2)*(mu_k(2)*ETA21-ETA22)+
        Lmat(2,3)*Qmat(3,2)/(z23-zo3)*(mu_k(3)*ETA21-ETA22));
    HW(2,6)=HW(2,6)+OME(i)*(COL-inicio)/2*XJA1*F21/pi*real(Lmat(3,1)*Qmat(1,2)
        /(z11-zo1)*(mu_k(1)*ETA11-ETA12)+Lmat(3,2)*Qmat(2,2)/(z12-zo2)*(mu_k(2)
        *ETA11-ETA12)+Lmat(3,3)*Qmat(3,2)/(z13-zo3)*(mu_k(3)*ETA11-ETA12))+OME(
        i)*(final-COL)/2*XJA2*F22/pi*real(Lmat(3,1)*Qmat(1,2)/(z21-zo1)*(mu_k
        (1)*ETA21-ETA22)+Lmat(3,2)*Qmat(2,2)/(z22-zo2)*(mu_k(2)*ETA21-ETA22)+
        Lmat(3,3)*Qmat(3,2)/(z23-zo3)*(mu_k(3)*ETA21-ETA22));
    HW(2,7)=HW(2,7)+OME(i)*(COL-inicio)/2*XJA1*F31/pi*real(Lmat(1,1)*Qmat(1,2)
        /(z11-zo1)*(mu_k(1)*ETA11-ETA12)+Lmat(1,2)*Qmat(2,2)/(z12-zo2)*(mu_k(2)
        *ETA11-ETA12)+Lmat(1,3)*Qmat(3,2)/(z13-zo3)*(mu_k(3)*ETA11-ETA12))+OME(
        i)*(final-COL)/2*XJA2*F32/pi*real(Lmat(1,1)*Qmat(1,2)/(z21-zo1)*(mu_k
        (1)*ETA21-ETA22)+Lmat(1,2)*Qmat(2,2)/(z22-zo2)*(mu_k(2)*ETA21-ETA22)+
        Lmat(1,3)*Qmat(3,2)/(z23-zo3)*(mu_k(3)*ETA21-ETA22));
    HW(2,8)=HW(2,8)+OME(i)*(COL-inicio)/2*XJA1*F31/pi*real(Lmat(2,1)*Qmat(1,2)
        /(z11-zo1)*(mu_k(1)*ETA11-ETA12)+Lmat(2,2)*Qmat(2,2)/(z12-zo2)*(mu_k(2)
        *ETA11-ETA12)+Lmat(2,3)*Qmat(3,2)/(z13-zo3)*(mu_k(3)*ETA11-ETA12))+OME(
        i)*(final-COL)/2*XJA2*F32/pi*real(Lmat(2,1)*Qmat(1,2)/(z21-zo1)*(mu_k
        (1)*ETA21-ETA22)+Lmat(2,2)*Qmat(2,2)/(z22-zo2)*(mu_k(2)*ETA21-ETA22)+
        Lmat(2,3)*Qmat(3,2)/(z23-zo3)*(mu_k(3)*ETA21-ETA22));
    HW(2,9)=HW(2,9)+OME(i)*(COL-inicio)/2*XJA1*F31/pi*real(Lmat(3,1)*Qmat(1,2)
        /(z11-zo1)*(mu_k(1)*ETA11-ETA12)+Lmat(3,2)*Qmat(2,2)/(z12-zo2)*(mu_k(2)
        *ETA11-ETA12)+Lmat(3,3)*Qmat(3,2)/(z13-zo3)*(mu_k(3)*ETA11-ETA12))+OME(
        i)*(final-COL)/2*XJA2*F32/pi*real(Lmat(3,1)*Qmat(1,2)/(z21-zo1)*(mu_k
        (1)*ETA21-ETA22)+Lmat(3,2)*Qmat(2,2)/(z22-zo2)*(mu_k(2)*ETA21-ETA22)+
        Lmat(3,3)*Qmat(3,2)/(z23-zo3)*(mu_k(3)*ETA21-ETA22));
    HW(3,1)=HW(3,1)+OME(i)*(COL-inicio)/2*XJA1*F11/pi*real(Lmat(1,1)*Qmat(1,3)
        /(z11-zo1)*(mu_k(1)*ETA11-ETA12)+Lmat(1,2)*Qmat(2,3)/(z12-zo2)*(mu_k(2)
        *ETA11-ETA12)+Lmat(1,3)*Qmat(3,3)/(z13-zo3)*(mu_k(3)*ETA11-ETA12))+OME(
```

```
        i)*(final-COL)/2*XJA2*F12/pi*real(Lmat(1,1)*Qmat(1,3)/(z21-zo1)*(mu_k
        (1)*ETA21-ETA22)+Lmat(1,2)*Qmat(2,3)/(z22-zo2)*(mu_k(2)*ETA21-ETA22)+
        Lmat(1,3)*Qmat(3,3)/(z23-zo3)*(mu_k(3)*ETA21-ETA22));
    HW(3,2)=HW(3,2)+OME(i)*(COL-inicio)/2*XJA1*F11/pi*real(Lmat(2,1)*Qmat(1,3)
        /(z11-zo1)*(mu_k(1)*ETA11-ETA12)+Lmat(2,2)*Qmat(2,3)/(z12-zo2)*(mu_k(2)
        *ETA11-ETA12)+Lmat(2,3)*Qmat(3,3)/(z13-zo3)*(mu_k(3)*ETA11-ETA12))+OME(
        i)*(final-COL)/2*XJA2*F12/pi*real(Lmat(2,1)*Qmat(1,3)/(z21-zo1)*(mu_k
        (1)*ETA21-ETA22)+Lmat(2,2)*Qmat(2,3)/(z22-zo2)*(mu_k(2)*ETA21-ETA22)+
        Lmat(2,3)*Qmat(3,3)/(z23-zo3)*(mu_k(3)*ETA21-ETA22));
    HW(3,3)=HW(3,3)+OME(i)*(COL-inicio)/2*XJA1*F11/pi*real(Lmat(3,1)*Qmat(1,3)
        /(z11-zo1)*(mu_k(1)*ETA11-ETA12)+Lmat(3,2)*Qmat(2,3)/(z12-zo2)*(mu_k(2)
        *ETA11-ETA12)+Lmat(3,3)*Qmat(3,3)/(z13-zo3)*(mu_k(3)*ETA11-ETA12))+OME(
        i)*(final-COL)/2*XJA2*F12/pi*real(Lmat(3,1)*Qmat(1,3)/(z21-zo1)*(mu_k
        (1)*ETA21-ETA22)+Lmat(3,2)*Qmat(2,3)/(z22-zo2)*(mu_k(2)*ETA21-ETA22)+
        Lmat(3,3)*Qmat(3,3)/(z23-zo3)*(mu_k(3)*ETA21-ETA22));
    HW(3,4)=HW(3,4)+OME(i)*(COL-inicio)/2*XJA1*F21/pi*real(Lmat(1,1)*Qmat(1,3)
        /(z11-zo1)*(mu_k(1)*ETA11-ETA12)+Lmat(1,2)*Qmat(2,3)/(z12-zo2)*(mu_k(2)
        *ETA11-ETA12)+Lmat(1,3)*Qmat(3,3)/(z13-zo3)*(mu_k(3)*ETA11-ETA12))+OME(
        i)*(final-COL)/2*XJA2*F22/pi*real(Lmat(1,1)*Qmat(1,3)/(z21-zo1)*(mu_k
        (1)*ETA21-ETA22)+Lmat(1,2)*Qmat(2,3)/(z22-zo2)*(mu_k(2)*ETA21-ETA22)+
        Lmat(1,3)*Qmat(3,3)/(z23-zo3)*(mu_k(3)*ETA21-ETA22));
    HW(3,5)=HW(3,5)+OME(i)*(COL-inicio)/2*XJA1*F21/pi*real(Lmat(2,1)*Qmat(1,3)
        /(z11-zo1)*(mu_k(1)*ETA11-ETA12)+Lmat(2,2)*Qmat(2,3)/(z12-zo2)*(mu_k(2)
        *ETA11-ETA12)+Lmat(2,3)*Qmat(3,3)/(z13-zo3)*(mu_k(3)*ETA11-ETA12))+OME(
        i)*(final-COL)/2*XJA2*F22/pi*real(Lmat(2,1)*Qmat(1,3)/(z21-zo1)*(mu_k
        (1)*ETA21-ETA22)+Lmat(2,2)*Qmat(2,3)/(z22-zo2)*(mu_k(2)*ETA21-ETA22)+
        Lmat(2,3)*Qmat(3,3)/(z23-zo3)*(mu_k(3)*ETA21-ETA22));
    HW(3,6)=HW(3,6)+OME(i)*(COL-inicio)/2*XJA1*F21/pi*real(Lmat(3,1)*Qmat(1,3)
        /(z11-zo1)*(mu_k(1)*ETA11-ETA12)+Lmat(3,2)*Qmat(2,3)/(z12-zo2)*(mu_k(2)
        *ETA11-ETA12)+Lmat(3,3)*Qmat(3,3)/(z13-zo3)*(mu_k(3)*ETA11-ETA12))+OME(
        i)*(final-COL)/2*XJA2*F22/pi*real(Lmat(3,1)*Qmat(1,3)/(z21-zo1)*(mu_k
        (1)*ETA21-ETA22)+Lmat(3,2)*Qmat(2,3)/(z22-zo2)*(mu_k(2)*ETA21-ETA22)+
        Lmat(3,3)*Qmat(3,3)/(z23-zo3)*(mu_k(3)*ETA21-ETA22));
    HW(3,7)=HW(3,7)+OME(i)*(COL-inicio)/2*XJA1*F31/pi*real(Lmat(1,1)*Qmat(1,3)
        /(z11-zo1)*(mu_k(1)*ETA11-ETA12)+Lmat(1,2)*Qmat(2,3)/(z12-zo2)*(mu_k(2)
        *ETA11-ETA12)+Lmat(1,3)*Qmat(3,3)/(z13-zo3)*(mu_k(3)*ETA11-ETA12))+OME(
        i)*(final-COL)/2*XJA2*F32/pi*real(Lmat(1,1)*Qmat(1,3)/(z21-zo1)*(mu_k
        (1)*ETA21-ETA22)+Lmat(1,2)*Qmat(2,3)/(z22-zo2)*(mu_k(2)*ETA21-ETA22)+
        Lmat(1,3)*Qmat(3,3)/(z23-zo3)*(mu_k(3)*ETA21-ETA22));
    HW(3,8)=HW(3,8)+OME(i)*(COL-inicio)/2*XJA1*F31/pi*real(Lmat(2,1)*Qmat(1,3)
        /(z11-zo1)*(mu_k(1)*ETA11-ETA12)+Lmat(2,2)*Qmat(2,3)/(z12-zo2)*(mu_k(2)
        *ETA11-ETA12)+Lmat(2,3)*Qmat(3,3)/(z13-zo3)*(mu_k(3)*ETA11-ETA12))+OME(
        i)*(final-COL)/2*XJA2*F32/pi*real(Lmat(2,1)*Qmat(1,3)/(z21-zo1)*(mu_k
        (1)*ETA21-ETA22)+Lmat(2,2)*Qmat(2,3)/(z22-zo2)*(mu_k(2)*ETA21-ETA22)+
        Lmat(2,3)*Qmat(3,3)/(z23-zo3)*(mu_k(3)*ETA21-ETA22));
    HW(3,9)=HW(3,9)+OME(i)*(COL-inicio)/2*XJA1*F31/pi*real(Lmat(3,1)*Qmat(1,3)
        /(z11-zo1)*(mu_k(1)*ETA11-ETA12)+Lmat(3,2)*Qmat(2,3)/(z12-zo2)*(mu_k(2)
        *ETA11-ETA12)+Lmat(3,3)*Qmat(3,3)/(z13-zo3)*(mu_k(3)*ETA11-ETA12))+OME(
        i)*(final-COL)/2*XJA2*F32/pi*real(Lmat(3,1)*Qmat(1,3)/(z21-zo1)*(mu_k
        (1)*ETA21-ETA22)+Lmat(3,2)*Qmat(2,3)/(z22-zo2)*(mu_k(2)*ETA21-ETA22)+
        Lmat(3,3)*Qmat(3,3)/(z23-zo3)*(mu_k(3)*ETA21-ETA22));
    end
end

end
```

B.8 INTEREQIGABEMANI

```
function [SSOL,DSOL,U,T]=INTEREQIGABEMANI(N,X,Y,SOL,L,CX,CY,elRanges,knotVec,
    weight,NN,active,Afinal,Bfinal,KODEu,DFIu,KODE1v,DFI1v,Gs,Amat,Qmat,Lmat,mu
    _k,Cmat,collocPtsX,collocPtsY)

%This subroutine computes the values of stresses and displacements at
%internal points.


for n=1:NN
    n1{n}=size(elRanges{n});
end
NE=N/(2);
plot(CX,CY,'mx')

% Rearrange the input and the solution so we have all displacements and
% tractions in due order.

U=zeros(3*sum(N),1);
tam=size(KODE1v);
T=zeros(tam(2),1);

pos1=1;
pos2=1;
for k=1:3*sum(N)
    if KODEu(k)==1
        U(k)=DFIu(pos1);
        pos1=pos1+1;
    else
        U(k)=SOL(pos2);
        pos2=pos2+1;
    end
end

T(logical(KODE1v))=(DFI1v(logical(KODE1v)));
[~,~,aux]=unique(Gs(not(KODE1v)));
T(~KODE1v)=SOL(pos2+aux-1);
T=[0,-1,0,0,-1,0,0,-1,0,0,-1,0,0,-1,0,0,-1,0,SOL(111),0,0,SOL(112),0,0,SOL(113)
    ,0,0,SOL(113),0,0,SOL(114),0,0,SOL(115),0,0,SOL(115),0,0,SOL(116),0,0,SOL
    (117),0,0,SOL(117),0,0,SOL(118),0,0,SOL(119)
    ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    SOL(120),0,0,SOL(121),0,0,SOL(122),0,0,SOL(122),0,0,SOL(123),0,0,SOL(124)
    ,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

SSOL=zeros(1,6*L);
DSOL=zeros(1,3*L);
Afinal(:,end+1:end+3)=Afinal(:,1:3);
Afinal(end+1:end+3,:)=Afinal(1:3,:);
if L~=0
  %   D=Afinal\U;
   %  Q=Bfinal\T;
     for k=1:L
         pG=0;
         pH=0;
```

```
        for nn=1:NN
           for i=1:size(elRanges{nn},1)
               [HW,GW]=EXTINEQIGABEMANI(CX(k),CY(k),X{nn}(active{nn}(i)),Y{nn}(
                   active{nn}(i)),X{nn}(active{nn}(i)+1),Y{nn}(active{nn}(i)+1),
                   X{nn}(active{nn}(i)+2),Y{nn}(active{nn}(i)+2),elRanges{nn}(i
                   ,1),elRanges{nn}(i,2),knotVec{nn},weight{nn},active{nn}(i),
                   Qmat,Amat,Lmat,mu_k,collocPtsX{n}(1),collocPtsY{n}(1),i);
               [D11,D12,D13,D22,D23,D33,S11,S12,S13,S22,S23,S33]=
                   SIGMAEQIGABEMANI(CX(k),CY(k),X{nn}(active{nn}(i)),Y{nn}(
                   active{nn}(i)),X{nn}(active{nn}(i)+1),Y{nn}(active{nn}(i)+1),
                   X{nn}(active{nn}(i)+2),Y{nn}(active{nn}(i)+2),elRanges{nn}(i
                   ,1),elRanges{nn}(i,2),knotVec{nn},weight{nn},active{nn}(i),
                   Qmat,Amat,Lmat,mu_k,Cmat);
               AW=[Afinal(3*(active{nn}(i))-2:3*(active{nn}(i))+6,3*(active{nn
                   }(i))-2:3*(active{nn}(i))+6)];
               HW=HW/AW;
               GW=GW/AW;
               D11=D11/AW;
               D12=D12/AW;
               D13=D13/AW;
               D22=D22/AW;
               D23=D23/AW;
               D33=D33/AW;
               S11=S11/AW;
               S12=S12/AW;
               S13=S13/AW;
               S22=S22/AW;
               S23=S23/AW;
               S33=S33/AW;
               for j=1:9
                   if i==size(elRanges{nn},1) && j>=7
                     SSOL(6*k-5)=SSOL(6*k-5)+D11(j)*T(9*i-9+j+pG)-S11(j)*U(j+pH
                         -6);
                     SSOL(6*k-4)=SSOL(6*k-4)+D12(j)*T(9*i-9+j+pG)-S12(j)*U(j+pH
                         -6);
                     SSOL(6*k-3)=SSOL(6*k-3)+D13(j)*T(9*i-9+j+pG)-S13(j)*U(j+pH
                         -6);
                     SSOL(6*k-2)=SSOL(6*k-2)+D22(j)*T(9*i-9+j+pG)-S22(j)*U(j+pH
                         -6);
                     SSOL(6*k-1)=SSOL(6*k-1)+D23(j)*T(9*i-9+j+pG)-S23(j)*U(j+pH
                         -6);
                     SSOL(6*k)=SSOL(6*k)+D33(j)*T(9*i-9+j+pG)-S33(j)*U(j+pH-6);

                     DSOL(3*k-2)=DSOL(3*k-2)+GW(1,j)*T(9*i-9+j+pG)-HW(1,j)*U(j+
                         pH-6);
                     DSOL(3*k-1)=DSOL(3*k-1)+GW(2,j)*T(9*i-9+j+pG)-HW(2,j)*U(j+
                         pH-6);
                     DSOL(3*k)=DSOL(3*k)+GW(3,j)*T(9*i-9+j+pG)-HW(3,j)*U(j+pH-6)
                         ;
                   else
                     SSOL(6*k-5)=SSOL(6*k-5)+D11(j)*T(9*i-9+j+pG)-S11(j)*U((3*(
                         active{nn}(i))-3)+j+pH);
                     SSOL(6*k-4)=SSOL(6*k-4)+D12(j)*T(9*i-9+j+pG)-S12(j)*U((3*(
                         active{nn}(i))-3)+j+pH);
                     SSOL(6*k-3)=SSOL(6*k-3)+D13(j)*T(9*i-9+j+pG)-S13(j)*U((3*(
                         active{nn}(i))-3)+j+pH);
```

```
                    SSOL(6*k-2)=SSOL(6*k-2)+D22(j)*T(9*i-9+j+pG)-S22(j)*U((3*(
                        active{nn}(i))-3)+j+pH);
                    SSOL(6*k-1)=SSOL(6*k-1)+D23(j)*T(9*i-9+j+pG)-S23(j)*U((3*(
                        active{nn}(i))-3)+j+pH);
                    SSOL(6*k)=SSOL(6*k)+D33(j)*T(9*i-9+j+pG)-S33(j)*U((3*(
                        active{nn}(i))-3)+j+pH);

                    DSOL(3*k-2)=DSOL(3*k-2)+GW(1,j)*T(9*i-9+j+pG)-HW(1,j)*U
                        ((3*(active{nn}(i))-3)+j+pH);
                    DSOL(3*k-1)=DSOL(3*k-1)+GW(2,j)*T(9*i-9+j+pG)-HW(2,j)*U
                        ((3*(active{nn}(i))-3)+j+pH);
                    DSOL(3*k)=DSOL(3*k)+GW(3,j)*T(9*i-9+j+pG)-HW(3,j)*U((3*(
                        active{nn}(i))-3)+j+pH);
                  end
               end
            end
            if i==n1{nn}(1)
                  pG=pG+9*size(elRanges{nn},1);
                  pH=pH+3*N(nn);
            end
         end
     end
end

end
```

B.9 SIGMAEQIGABEMANI

```
function [D11,D12,D13,D22,D23,D33,S11,S12,S13,S22,S23,S33]=SIGMAEQIGABEMANI(XP,
    YP,X1,Y1,X2,Y2,X3,Y3,inicio,final,knotVec,weight,active,Qmat,Amat,Lmat,mu_k
    ,Cmat)

%This subroutine computes the values of the S and D matrices using Gauss
%quadrature in order to compute the stress at any internal point.

%RA=RADIUS.
%RD1,RD2,RDN=RADIUS DERIVATIVES.
%ETA1,ETA2=COMPONENTS OF THE UNIT NORMAL TO THE ELEMENT.
%XCO,YCO=INTEGRATION POINT ALONG THE ELEMENT.
%XJA=JACOBIAN.

GI=[0.9739065285,-0.9739065285,0.8650633665,-0.8650633665,0.6794095683,
-0.6794095683,0.4333953941,-0.4333953941,0.1488743389,-0.1488743389];
OME=[0.0666713443,0.0666713443,0.1494513491,0.1494513491,0.2190863625,
0.2190863625,0.2692667193,0.2692667193,0.2955242247,0.2955242247];
D11=zeros(1,9);
D12=zeros(1,9);
D13=zeros(1,9);
D22=zeros(1,9);
D23=zeros(1,9);
D33=zeros(1,9);
S11=zeros(1,9);
S12=zeros(1,9);
S13=zeros(1,9);
```

```
S22=zeros(1,9);
S23=zeros(1,9);
S33=zeros(1,9);
JAC=(final-inicio)/2;

% Transformation to complex plane

zo1=XP+YP*mu_k(1);
zo2=XP+YP*mu_k(2);
zo3=XP+YP*mu_k(3);

%Compute values of the shape functions at the integration points.

for i=1:10

    [F1,dR1]=NURBSbasis_matlab(active,2,(GI(i)*(final-inicio)+final+inicio)/2,
        knotVec,weight);
    [F2,dR2]=NURBSbasis_matlab(active+1,2,(GI(i)*(final-inicio)+final+inicio)
        /2,knotVec,weight);
    [F3,dR3]=NURBSbasis_matlab(active+2,2,(GI(i)*(final-inicio)+final+inicio)
        /2,knotVec,weight);

    %Compute geometrical parameters.

    XCO=X1*F1+X2*F2+X3*F3;
    YCO=Y1*F1+Y2*F2+Y3*F3;
    A=dR1*X1+dR2*X2+dR3*X3;
    B=dR1*Y1+dR2*Y2+dR3*Y3;
    XJA=sqrt(A^2+B^2);
    ETA1=B/XJA;
    ETA2=-A/XJA;

    % Transformation to complex plane

    z1=XCO+YCO*mu_k(1);
    z2=XCO+YCO*mu_k(2);
    z3=XCO+YCO*mu_k(3);

    %Compute D and S coefficients.

    %Compute D and S coefficients.

    U11_1=1/pi*real(Amat(1,1)*Qmat(1,1)/(z1-zo1)+Amat(1,2)*Qmat(2,1)/(z2-zo2)+
        Amat(1,3)*Qmat(3,1)/(z3-zo3));
    U11_2=1/pi*real(Amat(1,1)*Qmat(1,1)/(z1-zo1)*mu_k(1)+Amat(1,2)*Qmat(2,1)/(z
        2-zo2)*mu_k(2)+Amat(1,3)*Qmat(3,1)/(z3-zo3)*mu_k(3));
    U12_1=1/pi*real(Amat(2,1)*Qmat(1,1)/(z1-zo1)+Amat(2,2)*Qmat(2,1)/(z2-zo2)+
        Amat(2,3)*Qmat(3,1)/(z3-zo3));
    U12_2=1/pi*real(Amat(2,1)*Qmat(1,1)/(z1-zo1)*mu_k(1)+Amat(2,2)*Qmat(2,1)/(z
        2-zo2)*mu_k(2)+Amat(2,3)*Qmat(3,1)/(z3-zo3)*mu_k(3));
    U13_1=1/pi*real(Amat(3,1)*Qmat(1,1)/(z1-zo1)+Amat(3,2)*Qmat(2,1)/(z2-zo2)+
        Amat(3,3)*Qmat(3,1)/(z3-zo3));
    U13_2=1/pi*real(Amat(3,1)*Qmat(1,1)/(z1-zo1)*mu_k(1)+Amat(3,2)*Qmat(2,1)/(z
        2-zo2)*mu_k(2)+Amat(3,3)*Qmat(3,1)/(z3-zo3)*mu_k(3));
    U21_1=1/pi*real(Amat(1,1)*Qmat(1,2)/(z1-zo1)+Amat(1,2)*Qmat(2,2)/(z2-zo2)+
        Amat(1,3)*Qmat(3,2)/(z3-zo3));
```

```
U21_2=1/pi*real(Amat(1,1)*Qmat(1,2)/(z1-zo1)*mu_k(1)+Amat(1,2)*Qmat(2,2)/(z
    2-zo2)*mu_k(2)+Amat(1,3)*Qmat(3,2)/(z3-zo3)*mu_k(3));
U22_1=1/pi*real(Amat(2,1)*Qmat(1,2)/(z1-zo1)+Amat(2,2)*Qmat(2,2)/(z2-zo2)+
    Amat(2,3)*Qmat(3,2)/(z3-zo3));
U22_2=1/pi*real(Amat(2,1)*Qmat(1,2)/(z1-zo1)*mu_k(1)+Amat(2,2)*Qmat(2,2)/(z
    2-zo2)*mu_k(2)+Amat(2,3)*Qmat(3,2)/(z3-zo3)*mu_k(3));
U23_1=1/pi*real(Amat(3,1)*Qmat(1,2)/(z1-zo1)+Amat(3,2)*Qmat(2,2)/(z2-zo2)+
    Amat(3,3)*Qmat(3,2)/(z3-zo3));
U23_2=1/pi*real(Amat(3,1)*Qmat(1,2)/(z1-zo1)*mu_k(1)+Amat(3,2)*Qmat(2,2)/(z
    2-zo2)*mu_k(2)+Amat(3,3)*Qmat(3,2)/(z3-zo3)*mu_k(3));
U31_1=1/pi*real(Amat(1,1)*Qmat(1,3)/(z1-zo1)+Amat(1,2)*Qmat(2,3)/(z2-zo2)+
    Amat(1,3)*Qmat(3,3)/(z3-zo3));
U31_2=1/pi*real(Amat(1,1)*Qmat(1,3)/(z1-zo1)*mu_k(1)+Amat(1,2)*Qmat(2,3)/(z
    2-zo2)*mu_k(2)+Amat(1,3)*Qmat(3,3)/(z3-zo3)*mu_k(3));
U32_1=1/pi*real(Amat(2,1)*Qmat(1,3)/(z1-zo1)+Amat(2,2)*Qmat(2,3)/(z2-zo2)+
    Amat(2,3)*Qmat(3,3)/(z3-zo3));
U32_2=1/pi*real(Amat(2,1)*Qmat(1,3)/(z1-zo1)*mu_k(1)+Amat(2,2)*Qmat(2,3)/(z
    2-zo2)*mu_k(2)+Amat(2,3)*Qmat(3,3)/(z3-zo3)*mu_k(3));
U33_1=1/pi*real(Amat(3,1)*Qmat(1,3)/(z1-zo1)+Amat(3,2)*Qmat(2,3)/(z2-zo2)+
    Amat(3,3)*Qmat(3,3)/(z3-zo3));
U33_2=1/pi*real(Amat(3,1)*Qmat(1,3)/(z1-zo1)*mu_k(1)+Amat(3,2)*Qmat(2,3)/(z
    2-zo2)*mu_k(2)+Amat(3,3)*Qmat(3,3)/(z3-zo3)*mu_k(3));

P11_1=1/pi*real(Lmat(1,1)*Qmat(1,1)/(z1-zo1)^2*(mu_k(1)*ETA1-ETA2)+Lmat
    (1,2)*Qmat(2,1)/(z2-zo2)^2*(mu_k(2)*ETA1-ETA2)+Lmat(1,3)*Qmat(3,1)/(z3-
    zo3)^2*(mu_k(3)*ETA1-ETA2));
P11_2=1/pi*real(Lmat(1,1)*Qmat(1,1)/(z1-zo1)^2*(mu_k(1)*ETA1-ETA2)*mu_k(1)+
    Lmat(1,2)*Qmat(2,1)/(z2-zo2)^2*(mu_k(2)*ETA1-ETA2)*mu_k(2)+Lmat(1,3)*
    Qmat(3,1)/(z3-zo3)^2*(mu_k(3)*ETA1-ETA2)*mu_k(3));
P12_1=1/pi*real(Lmat(2,1)*Qmat(1,1)/(z1-zo1)^2*(mu_k(1)*ETA1-ETA2)+Lmat
    (2,2)*Qmat(2,1)/(z2-zo2)^2*(mu_k(2)*ETA1-ETA2)+Lmat(2,3)*Qmat(3,1)/(z3-
    zo3)^2*(mu_k(3)*ETA1-ETA2));
P12_2=1/pi*real(Lmat(2,1)*Qmat(1,1)/(z1-zo1)^2*(mu_k(1)*ETA1-ETA2)*mu_k(1)+
    Lmat(2,2)*Qmat(2,1)/(z2-zo2)^2*(mu_k(2)*ETA1-ETA2)*mu_k(2)+Lmat(2,3)*
    Qmat(3,1)/(z3-zo3)^2*(mu_k(3)*ETA1-ETA2)*mu_k(3));
P13_1=1/pi*real(Lmat(3,1)*Qmat(1,1)/(z1-zo1)^2*(mu_k(1)*ETA1-ETA2)+Lmat
    (3,2)*Qmat(2,1)/(z2-zo2)^2*(mu_k(2)*ETA1-ETA2)+Lmat(3,3)*Qmat(3,1)/(z3-
    zo3)^2*(mu_k(3)*ETA1-ETA2));
P13_2=1/pi*real(Lmat(3,1)*Qmat(1,1)/(z1-zo1)^2*(mu_k(1)*ETA1-ETA2)*mu_k(1)+
    Lmat(3,2)*Qmat(2,1)/(z2-zo2)^2*(mu_k(2)*ETA1-ETA2)*mu_k(2)+Lmat(3,3)*
    Qmat(3,1)/(z3-zo3)^2*(mu_k(3)*ETA1-ETA2)*mu_k(3));
P21_1=1/pi*real(Lmat(1,1)*Qmat(1,2)/(z1-zo1)^2*(mu_k(1)*ETA1-ETA2)+Lmat
    (1,2)*Qmat(2,2)/(z2-zo2)^2*(mu_k(2)*ETA1-ETA2)+Lmat(1,3)*Qmat(3,2)/(z3-
    zo3)^2*(mu_k(3)*ETA1-ETA2));
P21_2=1/pi*real(Lmat(1,1)*Qmat(1,2)/(z1-zo1)^2*(mu_k(1)*ETA1-ETA2)*mu_k(1)+
    Lmat(1,2)*Qmat(2,2)/(z2-zo2)^2*(mu_k(2)*ETA1-ETA2)*mu_k(2)+Lmat(1,3)*
    Qmat(3,2)/(z3-zo3)^2*(mu_k(3)*ETA1-ETA2)*mu_k(3));
P22_1=1/pi*real(Lmat(2,1)*Qmat(1,2)/(z1-zo1)^2*(mu_k(1)*ETA1-ETA2)+Lmat
    (2,2)*Qmat(2,2)/(z2-zo2)^2*(mu_k(2)*ETA1-ETA2)+Lmat(2,3)*Qmat(3,2)/(z3-
    zo3)^2*(mu_k(3)*ETA1-ETA2));
P22_2=1/pi*real(Lmat(2,1)*Qmat(1,2)/(z1-zo1)^2*(mu_k(1)*ETA1-ETA2)*mu_k(1)+
    Lmat(2,2)*Qmat(2,2)/(z2-zo2)^2*(mu_k(2)*ETA1-ETA2)*mu_k(2)+Lmat(2,3)*
    Qmat(3,2)/(z3-zo3)^2*(mu_k(3)*ETA1-ETA2)*mu_k(3));
P23_1=1/pi*real(Lmat(3,1)*Qmat(1,2)/(z1-zo1)^2*(mu_k(1)*ETA1-ETA2)+Lmat
    (3,2)*Qmat(2,2)/(z2-zo2)^2*(mu_k(2)*ETA1-ETA2)+Lmat(3,3)*Qmat(3,2)/(z3-
    zo3)^2*(mu_k(3)*ETA1-ETA2));
```

```
P23_2=1/pi*real(Lmat(3,1)*Qmat(1,2)/(z1-zo1)^2*(mu_k(1)*ETA1-ETA2)*mu_k(1)+
    Lmat(3,2)*Qmat(2,2)/(z2-zo2)^2*(mu_k(2)*ETA1-ETA2)*mu_k(2)+Lmat(3,3)*
    Qmat(3,2)/(z3-zo3)^2*(mu_k(3)*ETA1-ETA2)*mu_k(3));
P31_1=1/pi*real(Lmat(1,1)*Qmat(1,3)/(z1-zo1)^2*(mu_k(1)*ETA1-ETA2)+Lmat
    (1,2)*Qmat(2,3)/(z2-zo2)^2*(mu_k(2)*ETA1-ETA2)+Lmat(1,3)*Qmat(3,3)/(z3-
    zo3)^2*(mu_k(3)*ETA1-ETA2));
P31_2=1/pi*real(Lmat(1,1)*Qmat(1,3)/(z1-zo1)^2*(mu_k(1)*ETA1-ETA2)*mu_k(1)+
    Lmat(1,2)*Qmat(2,3)/(z2-zo2)^2*(mu_k(2)*ETA1-ETA2)*mu_k(2)+Lmat(1,3)*
    Qmat(3,3)/(z3-zo3)^2*(mu_k(3)*ETA1-ETA2)*mu_k(3));
P32_1=1/pi*real(Lmat(2,1)*Qmat(1,3)/(z1-zo1)^2*(mu_k(1)*ETA1-ETA2)+Lmat
    (2,2)*Qmat(2,3)/(z2-zo2)^2*(mu_k(2)*ETA1-ETA2)+Lmat(2,3)*Qmat(3,3)/(z3-
    zo3)^2*(mu_k(3)*ETA1-ETA2));
P32_2=1/pi*real(Lmat(2,1)*Qmat(1,3)/(z1-zo1)^2*(mu_k(1)*ETA1-ETA2)*mu_k(1)+
    Lmat(2,2)*Qmat(2,3)/(z2-zo2)^2*(mu_k(2)*ETA1-ETA2)*mu_k(2)+Lmat(2,3)*
    Qmat(3,3)/(z3-zo3)^2*(mu_k(3)*ETA1-ETA2)*mu_k(3));
P33_1=1/pi*real(Lmat(3,1)*Qmat(1,3)/(z1-zo1)^2*(mu_k(1)*ETA1-ETA2)+Lmat
    (3,2)*Qmat(2,3)/(z2-zo2)^2*(mu_k(2)*ETA1-ETA2)+Lmat(3,3)*Qmat(3,3)/(z3-
    zo3)^2*(mu_k(3)*ETA1-ETA2));
P33_2=1/pi*real(Lmat(3,1)*Qmat(1,3)/(z1-zo1)^2*(mu_k(1)*ETA1-ETA2)*mu_k(1)+
    Lmat(3,2)*Qmat(2,3)/(z2-zo2)^2*(mu_k(2)*ETA1-ETA2)*mu_k(2)+Lmat(3,3)*
    Qmat(3,3)/(z3-zo3)^2*(mu_k(3)*ETA1-ETA2)*mu_k(3));

S11(1)=S11(1)+OME(i)*XJA*F1*(Cmat(1,1)*P11_1+Cmat(1,6)*P11_2+Cmat(1,6)*P
    21_1+Cmat(1,2)*P21_2+Cmat(1,5)*P31_1+Cmat(1,4)*P31_2);
S11(2)=S11(2)+OME(i)*XJA*F1*(Cmat(1,1)*P12_1+Cmat(1,6)*P12_2+Cmat(1,6)*P
    22_1+Cmat(1,2)*P22_2+Cmat(1,5)*P32_1+Cmat(1,4)*P32_2);
S11(3)=S11(3)+OME(i)*XJA*F1*(Cmat(1,1)*P13_1+Cmat(1,6)*P13_2+Cmat(1,6)*P
    23_1+Cmat(1,2)*P23_2+Cmat(1,5)*P33_1+Cmat(1,4)*P33_2);
S11(4)=S11(4)+OME(i)*XJA*F2*(Cmat(1,1)*P11_1+Cmat(1,6)*P11_2+Cmat(1,6)*P
    21_1+Cmat(1,2)*P21_2+Cmat(1,5)*P31_1+Cmat(1,4)*P31_2);
S11(5)=S11(5)+OME(i)*XJA*F2*(Cmat(1,1)*P12_1+Cmat(1,6)*P12_2+Cmat(1,6)*P
    22_1+Cmat(1,2)*P22_2+Cmat(1,5)*P32_1+Cmat(1,4)*P32_2);
S11(6)=S11(6)+OME(i)*XJA*F2*(Cmat(1,1)*P13_1+Cmat(1,6)*P13_2+Cmat(1,6)*P
    23_1+Cmat(1,2)*P23_2+Cmat(1,5)*P33_1+Cmat(1,4)*P33_2);
S11(7)=S11(7)+OME(i)*XJA*F3*(Cmat(1,1)*P11_1+Cmat(1,6)*P11_2+Cmat(1,6)*P
    21_1+Cmat(1,2)*P21_2+Cmat(1,5)*P31_1+Cmat(1,4)*P31_2);
S11(8)=S11(8)+OME(i)*XJA*F3*(Cmat(1,1)*P12_1+Cmat(1,6)*P12_2+Cmat(1,6)*P
    22_1+Cmat(1,2)*P22_2+Cmat(1,5)*P32_1+Cmat(1,4)*P32_2);
S11(9)=S11(9)+OME(i)*XJA*F3*(Cmat(1,1)*P13_1+Cmat(1,6)*P13_2+Cmat(1,6)*P
    23_1+Cmat(1,2)*P23_2+Cmat(1,5)*P33_1+Cmat(1,4)*P33_2);

S12(1)=S12(1)+OME(i)*XJA*F1*(Cmat(6,1)*P11_1+Cmat(6,6)*P11_2+Cmat(6,6)*P
    21_1+Cmat(6,2)*P21_2+Cmat(6,5)*P31_1+Cmat(6,4)*P31_2);
S12(2)=S12(2)+OME(i)*XJA*F1*(Cmat(6,1)*P12_1+Cmat(6,6)*P12_2+Cmat(6,6)*P
    22_1+Cmat(6,2)*P22_2+Cmat(6,5)*P32_1+Cmat(6,4)*P32_2);
S12(3)=S12(3)+OME(i)*XJA*F1*(Cmat(6,1)*P13_1+Cmat(6,6)*P13_2+Cmat(6,6)*P
    23_1+Cmat(6,2)*P23_2+Cmat(6,5)*P33_1+Cmat(6,4)*P33_2);
S12(4)=S12(4)+OME(i)*XJA*F2*(Cmat(6,1)*P11_1+Cmat(6,6)*P11_2+Cmat(6,6)*P
    21_1+Cmat(6,2)*P21_2+Cmat(6,5)*P31_1+Cmat(6,4)*P31_2);
S12(5)=S12(5)+OME(i)*XJA*F2*(Cmat(6,1)*P12_1+Cmat(6,6)*P12_2+Cmat(6,6)*P
    22_1+Cmat(6,2)*P22_2+Cmat(6,5)*P32_1+Cmat(6,4)*P32_2);
S12(6)=S12(6)+OME(i)*XJA*F3*(Cmat(6,1)*P13_1+Cmat(6,6)*P13_2+Cmat(6,6)*P
    23_1+Cmat(6,2)*P23_2+Cmat(6,5)*P33_1+Cmat(6,4)*P33_2);
S12(7)=S12(7)+OME(i)*XJA*F3*(Cmat(6,1)*P11_1+Cmat(6,6)*P11_2+Cmat(6,6)*P
    21_1+Cmat(6,2)*P21_2+Cmat(6,5)*P31_1+Cmat(6,4)*P31_2);
S12(8)=S12(8)+OME(i)*XJA*F3*(Cmat(6,1)*P12_1+Cmat(6,6)*P12_2+Cmat(6,6)*P
    22_1+Cmat(6,2)*P22_2+Cmat(6,5)*P32_1+Cmat(6,4)*P32_2);
```

```
      S12(9)=S12(9)+OME(i)*XJA*F3*(Cmat(6,1)*P13_1+Cmat(6,6)*P13_2+Cmat(6,6)*P
          23_1+Cmat(6,2)*P23_2+Cmat(6,5)*P33_1+Cmat(6,4)*P33_2);

      S13(1)=S13(1)+OME(i)*XJA*F1*(Cmat(5,1)*P11_1+Cmat(5,6)*P11_2+Cmat(5,6)*P
          21_1+Cmat(5,2)*P21_2+Cmat(5,5)*P31_1+Cmat(5,4)*P31_2);
      S13(2)=S13(2)+OME(i)*XJA*F1*(Cmat(5,1)*P12_1+Cmat(5,6)*P12_2+Cmat(5,6)*P
          22_1+Cmat(5,2)*P22_2+Cmat(5,5)*P32_1+Cmat(5,4)*P32_2);
      S13(3)=S13(3)+OME(i)*XJA*F1*(Cmat(5,1)*P13_1+Cmat(5,6)*P13_2+Cmat(5,6)*P
          23_1+Cmat(5,2)*P23_2+Cmat(5,5)*P33_1+Cmat(5,4)*P33_2);
      S13(4)=S13(4)+OME(i)*XJA*F2*(Cmat(5,1)*P11_1+Cmat(5,6)*P11_2+Cmat(5,6)*P
          21_1+Cmat(5,2)*P21_2+Cmat(5,5)*P31_1+Cmat(5,4)*P31_2);
      S13(5)=S13(5)+OME(i)*XJA*F2*(Cmat(5,1)*P12_1+Cmat(5,6)*P12_2+Cmat(5,6)*P
          22_1+Cmat(5,2)*P22_2+Cmat(5,5)*P32_1+Cmat(5,4)*P32_2);
      S13(6)=S13(6)+OME(i)*XJA*F3*(Cmat(5,1)*P13_1+Cmat(5,6)*P13_2+Cmat(5,6)*P
          23_1+Cmat(5,2)*P23_2+Cmat(5,5)*P33_1+Cmat(5,4)*P33_2);
      S13(7)=S13(7)+OME(i)*XJA*F3*(Cmat(5,1)*P11_1+Cmat(5,6)*P11_2+Cmat(5,6)*P
          21_1+Cmat(5,2)*P21_2+Cmat(5,5)*P31_1+Cmat(5,4)*P31_2);
      S13(8)=S13(8)+OME(i)*XJA*F3*(Cmat(5,1)*P12_1+Cmat(5,6)*P12_2+Cmat(5,6)*P
          22_1+Cmat(5,2)*P22_2+Cmat(5,5)*P32_1+Cmat(5,4)*P32_2);
      S13(9)=S13(9)+OME(i)*XJA*F3*(Cmat(5,1)*P13_1+Cmat(5,6)*P13_2+Cmat(5,6)*P
          23_1+Cmat(5,2)*P23_2+Cmat(5,5)*P33_1+Cmat(5,4)*P33_2);

      S22(1)=S22(1)+OME(i)*XJA*F1*(Cmat(2,1)*P11_1+Cmat(2,6)*P11_2+Cmat(2,6)*P
          21_1+Cmat(2,2)*P21_2+Cmat(2,5)*P31_1+Cmat(2,4)*P31_2);
      S22(2)=S22(2)+OME(i)*XJA*F1*(Cmat(2,1)*P12_1+Cmat(2,6)*P12_2+Cmat(2,6)*P
          22_1+Cmat(2,2)*P22_2+Cmat(2,5)*P32_1+Cmat(2,4)*P32_2);
      S22(3)=S22(3)+OME(i)*XJA*F1*(Cmat(2,1)*P13_1+Cmat(2,6)*P13_2+Cmat(2,6)*P
          23_1+Cmat(2,2)*P23_2+Cmat(2,5)*P33_1+Cmat(2,4)*P33_2);
      S22(4)=S22(4)+OME(i)*XJA*F2*(Cmat(2,1)*P11_1+Cmat(2,6)*P11_2+Cmat(2,6)*P
          21_1+Cmat(2,2)*P21_2+Cmat(2,5)*P31_1+Cmat(2,4)*P31_2);
      S22(5)=S22(5)+OME(i)*XJA*F2*(Cmat(2,1)*P12_1+Cmat(2,6)*P12_2+Cmat(2,6)*P
          22_1+Cmat(2,2)*P22_2+Cmat(2,5)*P32_1+Cmat(2,4)*P32_2);
      S22(6)=S22(6)+OME(i)*XJA*F2*(Cmat(2,1)*P13_1+Cmat(2,6)*P13_2+Cmat(2,6)*P
          23_1+Cmat(2,2)*P23_2+Cmat(2,5)*P33_1+Cmat(2,4)*P33_2);
      S22(7)=S22(7)+OME(i)*XJA*F3*(Cmat(2,1)*P11_1+Cmat(2,6)*P11_2+Cmat(2,6)*P
          21_1+Cmat(2,2)*P21_2+Cmat(2,5)*P31_1+Cmat(2,4)*P31_2);
      S22(8)=S22(8)+OME(i)*XJA*F3*(Cmat(2,1)*P12_1+Cmat(2,6)*P12_2+Cmat(2,6)*P
          22_1+Cmat(2,2)*P22_2+Cmat(2,5)*P32_1+Cmat(2,4)*P32_2);
      S22(9)=S22(9)+OME(i)*XJA*F3*(Cmat(2,1)*P13_1+Cmat(2,6)*P13_2+Cmat(2,6)*P
          23_1+Cmat(2,2)*P23_2+Cmat(2,5)*P33_1+Cmat(2,4)*P33_2);

      S23(1)=S23(1)+OME(i)*XJA*F1*(Cmat(4,1)*P11_1+Cmat(4,6)*P11_2+Cmat(4,6)*P
          21_1+Cmat(4,2)*P21_2+Cmat(4,5)*P31_1+Cmat(4,4)*P31_2);
      S23(2)=S23(2)+OME(i)*XJA*F1*(Cmat(4,1)*P12_1+Cmat(4,6)*P12_2+Cmat(4,6)*P
          22_1+Cmat(4,2)*P22_2+Cmat(4,5)*P32_1+Cmat(4,4)*P32_2);
      S23(3)=S23(3)+OME(i)*XJA*F1*(Cmat(4,1)*P13_1+Cmat(4,6)*P13_2+Cmat(4,6)*P
          23_1+Cmat(4,2)*P23_2+Cmat(4,5)*P33_1+Cmat(4,4)*P33_2);
      S23(4)=S23(4)+OME(i)*XJA*F2*(Cmat(4,1)*P11_1+Cmat(4,6)*P11_2+Cmat(4,6)*P
          21_1+Cmat(4,2)*P21_2+Cmat(4,5)*P31_1+Cmat(4,4)*P31_2);
      S23(5)=S23(5)+OME(i)*XJA*F2*(Cmat(4,1)*P12_1+Cmat(4,6)*P12_2+Cmat(4,6)*P
          22_1+Cmat(4,2)*P22_2+Cmat(4,5)*P32_1+Cmat(4,4)*P32_2);
      S23(6)=S23(6)+OME(i)*XJA*F2*(Cmat(4,1)*P13_1+Cmat(4,6)*P13_2+Cmat(4,6)*P
          23_1+Cmat(4,2)*P23_2+Cmat(4,5)*P33_1+Cmat(4,4)*P33_2);
      S23(7)=S23(7)+OME(i)*XJA*F3*(Cmat(4,1)*P11_1+Cmat(4,6)*P11_2+Cmat(4,6)*P
          21_1+Cmat(4,2)*P21_2+Cmat(4,5)*P31_1+Cmat(4,4)*P31_2);
      S23(8)=S23(8)+OME(i)*XJA*F3*(Cmat(4,1)*P12_1+Cmat(4,6)*P12_2+Cmat(4,6)*P
          22_1+Cmat(4,2)*P22_2+Cmat(4,5)*P32_1+Cmat(4,4)*P32_2);
```

```
      S23(9)=S23(9)+OME(i)*XJA*F3*(Cmat(4,1)*P13_1+Cmat(4,6)*P13_2+Cmat(4,6)*P
          23_1+Cmat(4,2)*P23_2+Cmat(4,5)*P33_1+Cmat(4,4)*P33_2);

      S33(1)=S33(1)+OME(i)*XJA*F1*(Cmat(3,1)*P11_1+Cmat(3,6)*P11_2+Cmat(3,6)*P
          21_1+Cmat(3,2)*P21_2+Cmat(3,5)*P31_1+Cmat(3,4)*P31_2);
      S33(2)=S33(2)+OME(i)*XJA*F1*(Cmat(3,1)*P12_1+Cmat(3,6)*P12_2+Cmat(3,6)*P
          22_1+Cmat(3,2)*P22_2+Cmat(3,5)*P32_1+Cmat(3,4)*P32_2);
      S33(3)=S33(3)+OME(i)*XJA*F1*(Cmat(3,1)*P13_1+Cmat(3,6)*P13_2+Cmat(3,6)*P
          23_1+Cmat(3,2)*P23_2+Cmat(3,5)*P33_1+Cmat(3,4)*P33_2);
      S33(4)=S33(4)+OME(i)*XJA*F2*(Cmat(3,1)*P11_1+Cmat(3,6)*P11_2+Cmat(3,6)*P
          21_1+Cmat(3,2)*P21_2+Cmat(3,5)*P31_1+Cmat(3,4)*P31_2);
      S33(5)=S33(5)+OME(i)*XJA*F2*(Cmat(3,1)*P12_1+Cmat(3,6)*P12_2+Cmat(3,6)*P
          22_1+Cmat(3,2)*P22_2+Cmat(3,5)*P32_1+Cmat(3,4)*P32_2);
      S33(6)=S33(6)+OME(i)*XJA*F2*(Cmat(3,1)*P13_1+Cmat(3,6)*P13_2+Cmat(3,6)*P
          23_1+Cmat(3,2)*P23_2+Cmat(3,5)*P33_1+Cmat(3,4)*P33_2);
      S33(7)=S33(7)+OME(i)*XJA*F3*(Cmat(3,1)*P11_1+Cmat(3,6)*P11_2+Cmat(3,6)*P
          21_1+Cmat(3,2)*P21_2+Cmat(3,5)*P31_1+Cmat(3,4)*P31_2);
      S33(8)=S33(8)+OME(i)*XJA*F3*(Cmat(3,1)*P12_1+Cmat(3,6)*P12_2+Cmat(3,6)*P
          22_1+Cmat(3,2)*P22_2+Cmat(3,5)*P32_1+Cmat(3,4)*P32_2);
      S33(9)=S33(9)+OME(i)*XJA*F3*(Cmat(3,1)*P13_1+Cmat(3,6)*P13_2+Cmat(3,6)*P
          23_1+Cmat(3,2)*P23_2+Cmat(3,5)*P33_1+Cmat(3,4)*P33_2);

      D11(1)=D11(1)+OME(i)*XJA*F1*(Cmat(1,1)*U11_1+Cmat(1,6)*U11_2+Cmat(1,6)*U
          21_1+Cmat(1,2)*U21_2+Cmat(1,5)*U31_1+Cmat(1,4)*U31_2);
      D11(2)=D11(2)+OME(i)*XJA*F1*(Cmat(1,1)*U12_1+Cmat(1,6)*U12_2+Cmat(1,6)*U
          22_1+Cmat(1,2)*U22_2+Cmat(1,5)*U32_1+Cmat(1,4)*U32_2);
      D11(3)=D11(3)+OME(i)*XJA*F1*(Cmat(1,1)*U13_1+Cmat(1,6)*U13_2+Cmat(1,6)*U
          23_1+Cmat(1,2)*U23_2+Cmat(1,5)*U33_1+Cmat(1,4)*U33_2);
      D11(4)=D11(4)+OME(i)*XJA*F2*(Cmat(1,1)*U11_1+Cmat(1,6)*U11_2+Cmat(1,6)*U
          21_1+Cmat(1,2)*U21_2+Cmat(1,5)*U31_1+Cmat(1,4)*U31_2);
      D11(5)=D11(5)+OME(i)*XJA*F2*(Cmat(1,1)*U12_1+Cmat(1,6)*U12_2+Cmat(1,6)*U
          22_1+Cmat(1,2)*U22_2+Cmat(1,5)*U32_1+Cmat(1,4)*U32_2);
      D11(6)=D11(6)+OME(i)*XJA*F2*(Cmat(1,1)*U13_1+Cmat(1,6)*U13_2+Cmat(1,6)*U
          23_1+Cmat(1,2)*U23_2+Cmat(1,5)*U33_1+Cmat(1,4)*U33_2);
      D11(7)=D11(7)+OME(i)*XJA*F3*(Cmat(1,1)*U11_1+Cmat(1,6)*U11_2+Cmat(1,6)*U
          21_1+Cmat(1,2)*U21_2+Cmat(1,5)*U31_1+Cmat(1,4)*U31_2);
      D11(8)=D11(8)+OME(i)*XJA*F3*(Cmat(1,1)*U12_1+Cmat(1,6)*U12_2+Cmat(1,6)*U
          22_1+Cmat(1,2)*U22_2+Cmat(1,5)*U32_1+Cmat(1,4)*U32_2);
      D11(9)=D11(9)+OME(i)*XJA*F3*(Cmat(1,1)*U13_1+Cmat(1,6)*U13_2+Cmat(1,6)*U
          23_1+Cmat(1,2)*U23_2+Cmat(1,5)*U33_1+Cmat(1,4)*U33_2);

      D12(1)=D12(1)+OME(i)*XJA*F1*(Cmat(6,1)*U11_1+Cmat(6,6)*U11_2+Cmat(6,6)*U
          21_1+Cmat(6,2)*U21_2+Cmat(6,5)*U31_1+Cmat(6,4)*U31_2);
      D12(2)=D12(2)+OME(i)*XJA*F1*(Cmat(6,1)*U12_1+Cmat(6,6)*U12_2+Cmat(6,6)*U
          22_1+Cmat(6,2)*U22_2+Cmat(6,5)*U32_1+Cmat(6,4)*U32_2);
      D12(3)=D12(3)+OME(i)*XJA*F1*(Cmat(6,1)*U13_1+Cmat(6,6)*U13_2+Cmat(6,6)*U
          23_1+Cmat(6,2)*U23_2+Cmat(6,5)*U33_1+Cmat(6,4)*U33_2);
      D12(4)=D12(4)+OME(i)*XJA*F2*(Cmat(6,1)*U11_1+Cmat(6,6)*U11_2+Cmat(6,6)*U
          21_1+Cmat(6,2)*U21_2+Cmat(6,5)*U31_1+Cmat(6,4)*U31_2);
      D12(5)=D12(5)+OME(i)*XJA*F2*(Cmat(6,1)*U12_1+Cmat(6,6)*U12_2+Cmat(6,6)*U
          22_1+Cmat(6,2)*U22_2+Cmat(6,5)*U32_1+Cmat(6,4)*U32_2);
      D12(6)=D12(6)+OME(i)*XJA*F2*(Cmat(6,1)*U13_1+Cmat(6,6)*U13_2+Cmat(6,6)*U
          23_1+Cmat(6,2)*U23_2+Cmat(6,5)*U33_1+Cmat(6,4)*U33_2);
      D12(7)=D12(7)+OME(i)*XJA*F3*(Cmat(6,1)*U11_1+Cmat(6,6)*U11_2+Cmat(6,6)*U
          21_1+Cmat(6,2)*U21_2+Cmat(6,5)*U31_1+Cmat(6,4)*U31_2);
      D12(8)=D12(8)+OME(i)*XJA*F3*(Cmat(6,1)*U12_1+Cmat(6,6)*U12_2+Cmat(6,6)*U
          22_1+Cmat(6,2)*U22_2+Cmat(6,5)*U32_1+Cmat(6,4)*U32_2);
```

```
      D12(9)=D12(9)+OME(i)*XJA*F3*(Cmat(6,1)*U13_1+Cmat(6,6)*U13_2+Cmat(6,6)*U
          23_1+Cmat(6,2)*U23_2+Cmat(6,5)*U33_1+Cmat(6,4)*U33_2);

      D13(1)=D13(1)+OME(i)*XJA*F1*(Cmat(5,1)*U11_1+Cmat(5,6)*U11_2+Cmat(5,6)*U
          21_1+Cmat(5,2)*U21_2+Cmat(5,5)*U31_1+Cmat(5,4)*U31_2);
      D13(2)=D13(2)+OME(i)*XJA*F1*(Cmat(5,1)*U12_1+Cmat(5,6)*U12_2+Cmat(5,6)*U
          22_1+Cmat(5,2)*U22_2+Cmat(5,5)*U32_1+Cmat(5,4)*U32_2);
      D13(3)=D13(3)+OME(i)*XJA*F1*(Cmat(5,1)*U13_1+Cmat(5,6)*U13_2+Cmat(5,6)*U
          23_1+Cmat(5,2)*U23_2+Cmat(5,5)*U33_1+Cmat(5,4)*U33_2);
      D13(4)=D13(4)+OME(i)*XJA*F2*(Cmat(5,1)*U11_1+Cmat(5,6)*U11_2+Cmat(5,6)*U
          21_1+Cmat(5,2)*U21_2+Cmat(5,5)*U31_1+Cmat(5,4)*U31_2);
      D13(5)=D13(5)+OME(i)*XJA*F2*(Cmat(5,1)*U12_1+Cmat(5,6)*U12_2+Cmat(5,6)*U
          22_1+Cmat(5,2)*U22_2+Cmat(5,5)*U32_1+Cmat(5,4)*U32_2);
      D13(6)=D13(6)+OME(i)*XJA*F2*(Cmat(5,1)*U13_1+Cmat(5,6)*U13_2+Cmat(5,6)*U
          23_1+Cmat(5,2)*U23_2+Cmat(5,5)*U33_1+Cmat(5,4)*U33_2);
      D13(7)=D13(7)+OME(i)*XJA*F3*(Cmat(5,1)*U11_1+Cmat(5,6)*U11_2+Cmat(5,6)*U
          21_1+Cmat(5,2)*U21_2+Cmat(5,5)*U31_1+Cmat(5,4)*U31_2);
      D13(8)=D13(8)+OME(i)*XJA*F3*(Cmat(5,1)*U12_1+Cmat(5,6)*U12_2+Cmat(5,6)*U
          22_1+Cmat(5,2)*U22_2+Cmat(5,5)*U32_1+Cmat(5,4)*U32_2);
      D13(9)=D13(9)+OME(i)*XJA*F3*(Cmat(5,1)*U13_1+Cmat(5,6)*U13_2+Cmat(5,6)*U
          23_1+Cmat(5,2)*U23_2+Cmat(5,5)*U33_1+Cmat(5,4)*U33_2);

      D22(1)=D22(1)+OME(i)*XJA*F1*(Cmat(2,1)*U11_1+Cmat(2,6)*U11_2+Cmat(2,6)*U
          21_1+Cmat(2,2)*U21_2+Cmat(2,5)*U31_1+Cmat(2,4)*U31_2);
      D22(2)=D22(2)+OME(i)*XJA*F1*(Cmat(2,1)*U12_1+Cmat(2,6)*U12_2+Cmat(2,6)*U
          22_1+Cmat(2,2)*U22_2+Cmat(2,5)*U32_1+Cmat(2,4)*U32_2);
      D22(3)=D22(3)+OME(i)*XJA*F1*(Cmat(2,1)*U13_1+Cmat(2,6)*U13_2+Cmat(2,6)*U
          23_1+Cmat(2,2)*U23_2+Cmat(2,5)*U33_1+Cmat(2,4)*U33_2);
      D22(4)=D22(4)+OME(i)*XJA*F2*(Cmat(2,1)*U11_1+Cmat(2,6)*U11_2+Cmat(2,6)*U
          21_1+Cmat(2,2)*U21_2+Cmat(2,5)*U31_1+Cmat(2,4)*U31_2);
      D22(5)=D22(5)+OME(i)*XJA*F2*(Cmat(2,1)*U12_1+Cmat(2,6)*U12_2+Cmat(2,6)*U
          22_1+Cmat(2,2)*U22_2+Cmat(2,5)*U32_1+Cmat(2,4)*U32_2);
      D22(6)=D22(6)+OME(i)*XJA*F2*(Cmat(2,1)*U13_1+Cmat(2,6)*U13_2+Cmat(2,6)*U
          23_1+Cmat(2,2)*U23_2+Cmat(2,5)*U33_1+Cmat(2,4)*U33_2);
      D22(7)=D22(7)+OME(i)*XJA*F3*(Cmat(2,1)*U11_1+Cmat(2,6)*U11_2+Cmat(2,6)*U
          21_1+Cmat(2,2)*U21_2+Cmat(2,5)*U31_1+Cmat(2,4)*U31_2);
      D22(8)=D22(8)+OME(i)*XJA*F3*(Cmat(2,1)*U12_1+Cmat(2,6)*U12_2+Cmat(2,6)*U
          22_1+Cmat(2,2)*U22_2+Cmat(2,5)*U32_1+Cmat(2,4)*U32_2);
      D22(9)=D22(9)+OME(i)*XJA*F3*(Cmat(2,1)*U13_1+Cmat(2,6)*U13_2+Cmat(2,6)*U
          23_1+Cmat(2,2)*U23_2+Cmat(2,5)*U33_1+Cmat(2,4)*U33_2);

      D23(1)=D23(1)+OME(i)*XJA*F1*(Cmat(4,1)*U11_1+Cmat(4,6)*U11_2+Cmat(4,6)*U
          21_1+Cmat(4,2)*U21_2+Cmat(4,5)*U31_1+Cmat(4,4)*U31_2);
      D23(2)=D23(2)+OME(i)*XJA*F1*(Cmat(4,1)*U12_1+Cmat(4,6)*U12_2+Cmat(4,6)*U
          22_1+Cmat(4,2)*U22_2+Cmat(4,5)*U32_1+Cmat(4,4)*U32_2);
      D23(3)=D23(3)+OME(i)*XJA*F1*(Cmat(4,1)*U13_1+Cmat(4,6)*U13_2+Cmat(4,6)*U
          23_1+Cmat(4,2)*U23_2+Cmat(4,5)*U33_1+Cmat(4,4)*U33_2);
      D23(4)=D23(4)+OME(i)*XJA*F2*(Cmat(4,1)*U11_1+Cmat(4,6)*U11_2+Cmat(4,6)*U
          21_1+Cmat(4,2)*U21_2+Cmat(4,5)*U31_1+Cmat(4,4)*U31_2);
      D23(5)=D23(5)+OME(i)*XJA*F2*(Cmat(4,1)*U12_1+Cmat(4,6)*U12_2+Cmat(4,6)*U
          22_1+Cmat(4,2)*U22_2+Cmat(4,5)*U32_1+Cmat(4,4)*U32_2);
      D23(6)=D23(6)+OME(i)*XJA*F2*(Cmat(4,1)*U13_1+Cmat(4,6)*U13_2+Cmat(4,6)*U
          23_1+Cmat(4,2)*U23_2+Cmat(4,5)*U33_1+Cmat(4,4)*U33_2);
      D23(7)=D23(7)+OME(i)*XJA*F3*(Cmat(4,1)*U11_1+Cmat(4,6)*U11_2+Cmat(4,6)*U
          21_1+Cmat(4,2)*U21_2+Cmat(4,5)*U31_1+Cmat(4,4)*U31_2);
      D23(8)=D23(8)+OME(i)*XJA*F3*(Cmat(4,1)*U12_1+Cmat(4,6)*U12_2+Cmat(4,6)*U
          22_1+Cmat(4,2)*U22_2+Cmat(4,5)*U32_1+Cmat(4,4)*U32_2);
```

```
    D23(9)=D23(9)+OME(i)*XJA*F3*(Cmat(4,1)*U13_1+Cmat(4,6)*U13_2+Cmat(4,6)*U
        23_1+Cmat(4,2)*U23_2+Cmat(4,5)*U33_1+Cmat(4,4)*U33_2);

    D33(1)=D33(1)+OME(i)*XJA*F1*(Cmat(3,1)*U11_1+Cmat(3,6)*U11_2+Cmat(3,6)*U
        21_1+Cmat(3,2)*U21_2+Cmat(3,5)*U31_1+Cmat(3,4)*U31_2);
    D33(2)=D33(2)+OME(i)*XJA*F1*(Cmat(3,1)*U12_1+Cmat(3,6)*U12_2+Cmat(3,6)*U
        22_1+Cmat(3,2)*U22_2+Cmat(3,5)*U32_1+Cmat(3,4)*U32_2);
    D33(3)=D33(3)+OME(i)*XJA*F1*(Cmat(3,1)*U13_1+Cmat(3,6)*U13_2+Cmat(3,6)*U
        23_1+Cmat(3,2)*U23_2+Cmat(3,5)*U33_1+Cmat(3,4)*U33_2);
    D33(4)=D33(4)+OME(i)*XJA*F2*(Cmat(3,1)*U11_1+Cmat(3,6)*U11_2+Cmat(3,6)*U
        21_1+Cmat(3,2)*U21_2+Cmat(3,5)*U31_1+Cmat(3,4)*U31_2);
    D33(5)=D33(5)+OME(i)*XJA*F2*(Cmat(3,1)*U12_1+Cmat(3,6)*U12_2+Cmat(3,6)*U
        22_1+Cmat(3,2)*U22_2+Cmat(3,5)*U32_1+Cmat(3,4)*U32_2);
    D33(6)=D33(6)+OME(i)*XJA*F2*(Cmat(3,1)*U13_1+Cmat(3,6)*U13_2+Cmat(3,6)*U
        23_1+Cmat(3,2)*U23_2+Cmat(3,5)*U33_1+Cmat(3,4)*U33_2);
    D33(7)=D33(7)+OME(i)*XJA*F3*(Cmat(3,1)*U11_1+Cmat(3,6)*U11_2+Cmat(3,6)*U
        21_1+Cmat(3,2)*U21_2+Cmat(3,5)*U31_1+Cmat(3,4)*U31_2);
    D33(8)=D33(8)+OME(i)*XJA*F3*(Cmat(3,1)*U12_1+Cmat(3,6)*U12_2+Cmat(3,6)*U
        22_1+Cmat(3,2)*U22_2+Cmat(3,5)*U32_1+Cmat(3,4)*U32_2);
    D33(9)=D33(9)+OME(i)*XJA*F3*(Cmat(3,1)*U13_1+Cmat(3,6)*U13_2+Cmat(3,6)*U
        23_1+Cmat(3,2)*U23_2+Cmat(3,5)*U33_1+Cmat(3,4)*U33_2);
end

D11=D11*JAC;
D12=D12*JAC;
D13=D13*JAC;
D22=D22*JAC;
D23=D23*JAC;
D33=D33*JAC;
S11=S11*JAC;
S12=S12*JAC;
S13=S13*JAC;
S22=S22*JAC;
S23=S23*JAC;
S33=S33*JAC;

end
```

References

[1] B.A. Auld. *Acoustic Fields and Waves in Solids*. Number v. 1 in A Wiley-Interscience publication. Wiley, 1973.

[2] Y. Bazilevs, V.M. Calo, J.A. Cottrell, J.A. Evans, T.J.R. Hughes, S. Lipton, M.A. Scott, and T.W. Sederberg. Isogeometric analysis using t-splines. *Computer Methods in Applied Mechanics and Engineering*, 199(5):229 – 263, 2010. Computational Geometry and Analysis.

[3] Y. Bazilevs, J.A. Cottrell, and T.J.R. Hughes. *Isogeometric Analysis*. WILEY, Singapore, 2009.

[4] C.A. Brebbia and J. Domínguez. Boundary elements. [accessed: 1st july 2015]. http://www.boundaryelements.com/index.php.

[5] C.A. Brebbia and J. Dominguez. *Boundary Elements An Introductory Course*. WITPRESS, Southampton, 1996.

[6] J.C. Marín A.Barroso F. París, J.Cañas. *Introducción al análisis y diseño con materiales compuestos.* Universidad de Sevilla. Escuela Técnica Superior de Ingenieros Industriales, 2008.

[7] F. (2005) García Sánchez. Estudio numérico de problemas de fractura en materiales anisótropos elásticos y piezoeléctricos.

[8] M. Guiggiani and P. Casalini. Direct computation of cauchy principal value integrals in advanced boundary elements. *Journal for Numerical Methods in Engineering*, 1987; 24: 1711-1720.

[9] R.W. Johnson. Higher order b-spline collocation at the greville abcissae. *Applied Numerical Mathematics*, 2005; 52: 63-75.

[10] Shivi Kesarwani. Polymer composites in aviation sector. *International Journal of Engineering Research and Technology*, V6, 06 2017.

[11] Fred Kocks, Carlos Tomé, and H-R Wenk. *Texture and Anisotropy. Preferred Orientations in Polycrystals and Their Effect on Material Properties*. 01 2000.

[12] Enayat Mahajerin and David L. Sikarskie. Boundary element study of a loaded hole in an orthotropic plate. *Journal of Composite Materials*, 20(4):375–389, 1986.

[13] V. Mallardo, E. Roucco, J.Trevelyan, and T.Rabczuk. An improved isogeometric boundary element method approach in two dimensional elastostatic. *CMES*, 2014; 102: 373-391.

[14] V. Mantič and F. París. Existence and evaluation of the two free terms in the hypersingular boundary integral equation of potential theory. *Engineering Analysis with Boundary Elements*, 16(3):253 – 260, 1995.

[15] V. Mantič and F. París. Explicit formulae of the integral kernels and c-matrix in the 2d somigliana identity for orthotropic materials. *Engineering Analysis with Boundary Elements*, 15(3):283 – 288, 1995.

[16] V. Mantič and F. París. Advanced formulation of the bies for elastic anisotropic materials in 2d. *Advances in Boundary Element Techniques V*, (3):57 – 62, 2004.

[17] J.J. Pérez-Gavilán. Introducción a los elementos de frontera. 2006. [accessed: 1st july 2015].

[18] B. Simpson. Isogeometric-bem. 2012. [accessed: 14th september 2020]. https://github.com/bobbiesimpson/isogeometric-bem.

[19] R.N. Simpson, J. Trevelyan, S.P.A. Bordas, and T. Rabczuk. A two-dimensional isogeometric boundary element method for elastostactic anaylisis. *Computer Methods in Applied Mechanics and Engineering*, 2012; 209-212: 87-100.

[20] T.C.T.Ting. *Anisotropic Elasticity Theory and Applications*. 1996.

[21] Ping Wang, Jinlan Xu, Jiansong Deng, and Falai Chen. Adaptive isogeometric analysis using rational pht-splines. *Computer-Aided Design*, 43(11):1438 – 1448, 2011. Solid and Physical Modeling 2011.