UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

FINAL DEGREE PROJECT

Degree in Biomedical Engineering

DEFORMATION AND TENSION ANALISYS DURING EMBRYONIC DEVELOPMENT

Memory and Annexes

Author: Alejandro Brugarolas Roca
Director: Jose Javier Muñoz Romero

Call: June, 2017

Resum

Aquest document tracta sobre el procés, a partir d'imatges microscòpiques tridimensionals, de les deformacions cel·lulars d'un embrió de mosca *Drosphila Melanogaster* durant el seu desenvolupament. També s'exposarà un programa amb el que calcular les tensions amb un codi d'elements finits a partir d'un model de material lineal.

Per tal de dur a terme aquest estudi, es farà ús del programa Matlab, amb el que es crearà un codi en el qual s'introduiran les dades de l'experiment i se'n estudiaran les tensions i les deformacions de les cèl·lules afectades. L'estudi es durà a terme a través de dos models: un d'ells centrat als centres de les cèl·lules i l'altre centrat als vèrtexs d'aquestes.

Amb els seus pertinents processats, s'obtindran una sèrie de fitxers que, al ser analitzats amb una eina de visualització tridimensional, oferiran una representació d'ambdós models i l'evolució de les seves pertinents malles. Aquest projecte conclourà amb una apreciació sobre l'eficàcia, tant de la aproximació bidimensional com la de la tridimensional.

Resumen

Este documento trata sobre el proceso, a través de imágenes microscópicas tridimensionales, de las deformaciones celulares de un embrión de mosca *Drosphila Melanogaster* durante su desarrollo. También se expondrá un programa con el que calcular las tensiones con un código de elementos finitos a partir de un modelo de material lineal.

Para llevar a cabo este estudio, se hará uso del programa Matlab, con el que se creará un código en el que se introducirán los datos del experimento y se estudiarán las tensiones y las deformaciones de las células afectadas. El estudio se llevará a cabo a través de dos modelos: uno de ellos centrado en los centros de las células y el otro centrado en los vértices de estas.

Con sus pertinentes procesados, se obtendrán una serie de documentos que, al ser analizados con una herramienta de visualización tridimensional, ofrecerán una representación de ambos modelos y la evolución de sus pertinentes mallas. Este proyecto concluirá con una apreciación sobre la eficacia, tanto de la aproximación bidimensional como la de la tridimensional.

Abstract

The aim of this document is to introduce the reader into the process, through microscopic three-dimensional images, of the cellular embryonic deformation of the *Drosphila Melanogaster* fly through its development. It will also be exposed a program with which it will be possible to calculate the tensions through a code of finite elements from a model of lineal material.

In order to carry this study, it will be used the Matlab program, with which a code will be created where the experimental data will be entered and the tension and the deformation of the affected cells will be calculated. The study will be carried through two models: one of them will be a cell-centered model while the other one will be a vertex model.

With the pertinent procedures, it will be obtained a series of documents that, when analyzed with a tridimensional visualization tool, will offer a representation of both models and the evolution for each of their meshes. This project will conclude with an appreciation on the efficiency of both two-dimensional and three-dimensional approaches.

Acknowledgements

First, I would like to express my most sincere gratitude to my tutor Jose Muñoz, for his help, patience and guidance through every step in this project. I also need to thank my family and friends, specially Guillem Balcells, Clara Elias, Marta Lana and Bet Maristany, who supported and helped me in every little doubt I had during these months.

Index

RESUM I

RESUMEN I

ABSTRACT II

ACKNOWLEDGEMENTS III

INDEX OF FIGURES VII

INDEX OF TABLES VIII

NOMENCLATURE IX

1. PREFACE 1

1.1. Motivation 1

1.2. Previous requirements 1

2. INTRODUCTION 3

2.1. State of the art 3

2.2. Project origin 3

2.3. Aim of the project 4

2.4. Scope of the project 4

2.5. The spring system 5

2.6. The embryo: Drosphila melanogaster 7

3. MECHANICAL MODEL 9

3.1. The cell-centered model 9

3.2. The vertex model 9

3.3. The hybrid model 10

4. METHODOLOGY 11

4.1. The Main program 12

4.2. The Boundary Filter function 12

4.2.1. Aspect Ratio Calculus 14

4.3. The Set Left Right function 20

4.4. The Voronoi Interpol function 22

4.5. The Output VTK function 23

4.5.1. VTK results comprehension 23

5. RESULTS 26

6. BUDGET 29

7. CONCLUSIONS AND FUTURE WORK 32

8. BIBLIOGRAPHY 33

8.1. Bibliographic References 33

8.2. Consulted Bibliography 33

ANNEX A 35

A.1. The Main program 35

A.2. The Geo function 37

A.3. The Boundary Filter Function 38

A.4. The Set Left Right function 40

A.5. The Voronoi Interpol function 42

A.6. The Voronoi function 43

Index of Figures

Figure 2.1. Spring symbolization..... 5

Figure 2.2. Dashpot symbolization.....5

Figure 2.3. Kelvin-Voigt system representation..... 5

Figure 2.4. Maxwell system representation..... 5

Figure 2.5. Active lengthening model system representation.....6

Figure 2.6. Drosphila melanogaster's egg..... 7

Figure 2.7. The Drosphila melanogaster.....8

Figure 3.1. Representation of the hybrid model..... 10

Figure 4.1. Scheme of the different steps that will take place during the process..... 11

Figure 4.2. Representation of a 2x2x2 network and a 3x3x3 network.....12

Figure 4.3. Example of internal and external elements in the mesh..... 13

Figure 4.4. Schematic view of the filtering process of Delaunay triangulation..... 15

Figure 4.5. Representation of a tetrahedron with its defining nodes.....16

Figure 4.6. Inradius and circumradius of a tetrahedron.....17

Figure 4.7. Evolution of a tetrahedron as one node moves away.....18

Figure 4.8. Comparison between both of the ratio calculus as distance increases..... 19

Figure 4.9. Representation of the possible cases in nT – tolerance comparison..... 21

Figure 4.10. Representation of a vertex and its coordinates..... 22

Figure 4.11. Parts of the VTK data file format..... 24

Figure 4.12. Example of a resultant VTK file..... 25

Figure 5.1. Representation of the results in a 3x3x3 network mesh, at the initial instant.....26

Figure 5.2. Sample geometry and stress evolution through time in a network mesh.....27

Figure 5.3. Illustration of the shrinking of the sample.....28

Figure 5.4. Illustration of the elongation of the sample..... 29

Index of Tables

Table 2.1. Drosphila melanogaster's taxonomy…… 7

Table 4.1. Classification of internal and external elements in Figure 4.2……14

Table 4.2. Linear cell types found in VTK……24

Table 6.1. Cost of the hardware required in the project……29

Table 6.2. Cost of the software required in the project…… 29

Table 6.3. Cost of the energy consumed by the devices……30

Table 6.4. Cost of the personnel required for the project…… 31

Table 6.5. Summary of the project cost…… 31

Nomenclature

B^{ij}	Bar element uniting x^i and x^j nodes. Section 3.1.
K	Spring constant Young modulus. Eq. 2.1.
l	Minimum edge length of a tetrahedron. Eq. 4.12.
l_{1-2}	Bar uniting two boundary nodes. Eq. 4.13.
l_{1-2-3}	Face uniting three boundary nodes. Eq. 4.16.
L	Maximum edge length of a tetrahedron. Eq. 4.12.
N_D	Total number of edges. Section 3.1.
N_F	Total number of faces. Section 3.1.
N_i	Number of vertices surrounding node i. Section 3.2.
N_{nodes}	Total number of nodes. Section 3.1.
nT	Normal of a boundary element. Eq. 4.15.
N_{tet}	Total number of tetrahedra in the nodal network. Section 3.1.
$p^i(\xi^I)$	Shape function defining vertex positions. Eq. 4.19.
r	Inradius of a tetrahedron. Eq. 4.8.
R	Circumradius of a tetrahedron. Eq. 4.9.
$\mathcal{T}^I$	Tetrahedron where vertex I is located. Section 3.1.
t_n	Instant of time. Section 3.1.
T_n	Connectivity of nodal network at time t_n. Section 3.1.
w_{lL}	Aspect ratio of a tetrahedron by the lengths ratio. Eq. 4.12.
w_{rR}	Aspect ratio of a tetrahedron by the radius ratio. Eq. 4.10.
x^i	Position of node i. Section 3.1.
X_n	List of nodal positions at time t_n. Section 3.1.
y^I	Position of vertex I. Section 3.2.
Δt	Time increment. Section 3.1.
γ	Remodeling rate of a material. Eq. 2.7.
ξ^I	Local coordinate of vertex I in triangle $\mathcal{T}^I$. Section 3.2.
ε	Strain. Section 2.5.
ε^e	Strain in a spring. Eq. 2.1.
ε^v	Strain in a dashpot. Eq. 2.2.
ε^ε	Elastic strain. Eq. 2.7.
η	Coefficient of viscosity. Eq. 2.2.
σ	Stress. Section 2.5.
σ^e	Stress in a spring. Eq. 2.1.
σ^v	Stress in a dashpot. Eq. 2.2.

1. Preface

1.1. Motivation

The great interest developed through the Mechanical Systems and Mechanics of Biological Structures subjects, added to the fact of being able to put into practice the knowledge acquired in Matlab through the degree, were the main motivations for carrying out this project.

Moreover, the fact of participating in an already existing study, which is been carried out for years by several people, and being part of an ongoing research process were two decisive points in the choice of the project.

1.2. Previous requirements

For this project to be done, the Matlab program was very important, so it was needed to have a solid base of knowledge in the programming language in order to be able to make a good use of this tool. Also a previous introduction to the Paraview program [1] and the VTK language was needed to be done before starting the project itself.

It was also important to know about mechanical issues in order to understand what the different cells were subjected at every step of the process. For last, to have a good sense of spatial view was crucial in this project because otherwise it would sometimes result to be difficult to understand how the different models and the networks were evolving through the process.

2. Introduction

2.1. State of the art

Mechanical analysis of tissues in an embryonic state is a way of research that is constantly increasing, and which provides and offers new ways of studying biological issues or new procedures which could be useful for future treatments. In order to carry out these analysis, there have been used diverse numerical approaches. These can be classified in two groups. The first of them is the continuum approach, which constitutes the behavior of solids or fluids. The second one, is the cell-based model, with which the tissue elements can be studied with more detail.

The cell-based models can be described either by a cell-centered model or by a vertex model. The former establishes the forces between cell-centers while the latter describes the mechanical forces at the cell-cell junctions by the description of the epithelia given by a set of vertices. If put these two models together, it is obtained a hybrid model which provides the advantages from both previous approaches: both centers and cell-cell junctions interactions are described. In order of this model to be described, it is necessary for the mechanical forces to be specified, because the mesh will be defined by its mechanical equilibrium.

2.2. Project origin

This is a continuation of an already existing project which studies the wound healing in a *Drosphila melanogaster* fly wing which have been damaged during the embryonic state. This is a project that have been carried out for some years, and which has been developed by P. Mosaffa, A. Rodríguez-Ferran and J.J. Muñoz. It has been studied the way the hybrid cell-centered/vertex model can be applied in this specific study and how the tissue and the healing of the wound evolve through time. This study has made use of a Matlab code which represents the evolution of the sample through time.

They based their study in a two dimensional approach, assuming that the main forces acting to deform the cells are generated along the cell surfaces. This way the study is also simplified from a mechanical point of view. On the other hand, though, this assumption also brings inaccuracy to the process: real tissues need to be seen in a three dimensional way, because even being extremely narrow, these have some thickness. Therefore, although this is a good approach it is not completely true to reality.

Moreover, there are some processes such as wound healing that are originated by three dimensional phenomena. Most of the diseases found in tissues also undergo a three dimensional process when expanding or being destroyed, for example.

From this background surges the need of finding a way to study the same processes that have been studied in a two dimensional way through a three dimensional point of view. And, although the main idea is almost the same for any of the two dimensional points of view, the implementation process varies in its majority.

This project, therefore, can be considered as the begging of a new part of the main study, with which the three dimensional study will begin and could help in next research processes.

2.3. Aim of the project

The main objective of this project is to modify a given Matlab code which works with a two dimensional given data so it can also work with three dimensional data. In order to do this, every program and every function given should be inspected. In this revision, not only the small errors will be corrected or modified but also some functions will be needed to be rewritten at its all.

Once obtained the final code, the results will be extracted and then compared and/or complemented with the ones obtained in the two dimensional code through the Paraview program. Conclusions will be then extracted and studied with some Matlab tools as well. This will allow conclusions to be drawn and to check if the two dimensional code works the same way as the three dimensional supposing that one of the coordinates axis is not taken into account.

2.4. Scope of the project

In order to carry out this project, the Matlab program will be used, with which a code will be created where the experimental data will be entered and the tensions and the deformations of the affected cells will be calculated. The study will be carried through two different points of view. The first of them will be through cell-centered model, in which the center of each of the cells will be calculated and it will be performed a Delaunay triangulation in order to obtain these points connectivity. The second method will be studied through a vertex model: from each of the tetrahedra in the previous triangulation, it is obtained an associated cell vertex. These two results will be transcribed into the VTK format and then represented with the help of the Paraview program.

2.5. The spring system

All mechanical elements are defined in terms of their force/motion relation. When a load is applied to the system in order to determine the deformation suffered, the experiment is classified as a Neumann experiment. This type of experiments can be carried out with the help of two basic elements: the spring and the dashpot.

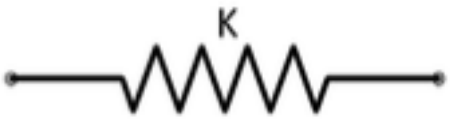


***Figure 2.1**. Spring symbolization.*

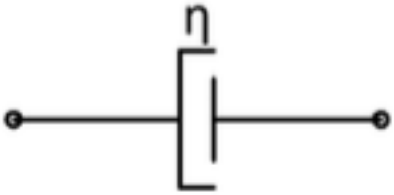


***Figure 2.2**. Dashpot symbolization.*

The relation between the stress σ and the strain ε in these two elements are given by the following equations, respectively:

$$\sigma^e = K * \varepsilon^e \quad \textbf{(Eq. 2.1)}$$

$$\sigma^v = \eta * \frac{d\varepsilon}{dt} = \eta * \varepsilon^v \quad \textbf{(Eq. 2.2)}$$

where K is the spring constant Young modulus and η is the coefficient of viscosity.

The Kelvin-Voigt and Maxwell models were designed to simulate viscoelasticity materials. In the Kelvin-Voigt model a spring element and dashpot element are connected in parallel, while in the Maxwell model these are connected in series.

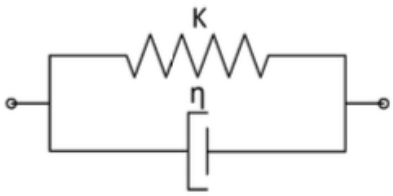


***Figure 2.3**. Kelvin-Voigt system representation.*

***Figure 2.4**. Maxwell system representation.*

The Kelvin-Voigt model is studied through the following relations,

$$\sigma = \sigma^e + \sigma^v \quad \textbf{(Eq. 2.3)}$$

$$\varepsilon = \varepsilon^e = \varepsilon^v \quad \textbf{(Eq. 2.4)}$$

while the Maxwell model is studied through

$$\sigma = \sigma^e = \sigma^v \quad \textbf{(Eq. 2.5)}$$

$$\varepsilon = \varepsilon^e + \varepsilon^v \quad \textbf{(Eq. 2.6)}$$

There is also an active lengthening model. This model imposes that the cell viscosity is also related to the cell activity. This model considers the cell viscosity as a combination of the cell network and as a consequence of the active response of the motor proteins.

This experiment will undergo a process of stretching through time. This is due to the actin filaments in the cytoskeleton, which suffer from a non-reversible lengthening and remodeling. This fact is due to the remodeling of the cross-links and a depolymerisation of the filaments. Therefore, the affirmation of the total length of the filaments, when subjected to zero loads at their ends to be proportional to the elastic strain is arisen.

$$\frac{\dot{L}}{L} = \gamma \varepsilon^{\varepsilon} \quad \textbf{(Eq. 2.7)}$$

where $\varepsilon^{\varepsilon}$ is the elastic strain and γ is the remodeling rate of the material.

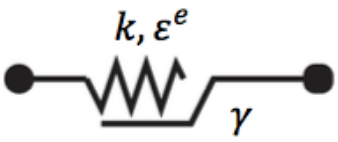


***Figure 2.5**. Active lengthening model system representation.*

2.6. The embryo: Drosphila melanogaster

Drosphila melanogaster is a species of fly that belongs to the *Drosophilidae* family. The use of this species in research was first proposed in the late 1880's by Charles W. Woodworth, and first used in 1910 by Thomas Hunt Morgan. In the time being, *Drosphila melanogaster* is universally used in different fields of research such as genetics, physiology, microbial pathogenesis and life history evolution. It is widely used because of its biological properties, which include its quick breed, the clutch of an elevated number of eggs and the possession of four pairs of chromosomes.

Kingdom	Animalia
Phylum	Arthropoda
Class	Insecta
Order	Diptera
Family	Drosophilidae
Genus	Drosophila
Subgenus	Sophophora
Species	Drosophila melanogaster

***Table 2.1**. Drosphila melanogaster's taxonomy.*

As said before, one important feature of the *Drosphila melanogaster* is the amount of eggs laid at once. Females lay around 400 eggs, which each of them hatch after 12-15 hours, depending on temperature of the breeding, becoming then a larva. The larva takes then about 4 days to grow and to give rise to the adult state of the fly. Its lifespan also depends on the temperature, varying between 7 to 50 days from the moment they are conceived.

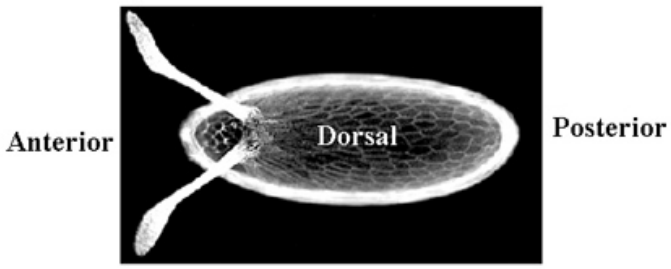


***Figure 2.6**. Drosphila melanogaster's egg. (Source: Quora.com)*

This species was one of the firsts organisms to be used in genetic analysis studies, and nowadays is one of the most used eukaryotic organisms, mainly because of its short generation time, which allows Scientifics to carry on studies in different generations in a short period time.

When it comes to similarity to humans, recent research estimated that about 60% of genes are conserved between the two species, a 75% of the human diseases have a match in the *Drosphila melanogaster* genome and a 50% of fly protein sequences have homologs in mammals. At the moment, the fly is being used for research into diverse neurodegenerative disorders and mechanisms underlying some type of disease.

***Figure 2.7**. The Drosphila melanogaster. (Source: YourGenome.org)*

3. Mechanical model

In order to carry out this project, it is needed to make use of two models merged into one so that the geometric locations and data needed for this study are found. These two approaches are cell-based models which enables the student to explicitly represent the junctional mechanics and the cellular nature of tissues [3, 4].

3.1. The cell-centered model

The proposed model is configured by the centers of each of the cells present in the mesh. These cell centers will be also named nodes. Each of these nodes will define a nodal position, which will be denoted as x^i. For this study, it will be supposed that the sample will have a constant number of cells N_{nodes}, whose cell-center positions will be defined as $X = \{x^i, \dots, x^{N_{nodes}}\}$ and the connectivity found between them will be referred as T which will conform a triangulation formed by N_{tet} tetrahedra $\mathcal{T}^{\mathrm{I}}$ and N_F faces and N_D edges.

This process will also be divided in different time steps, each of them separated with each other by a time increment Δt. Both, the set of nodal coordinates X_n and the connectivity T_n at the time t_n will be defined through the different steps of the process. Nodal position is determined through the mechanical equilibrium of the sample. A trimmed Delaunay triangulation will define the different connections found between the different nodes of the sample.

Each of these tetrahedra defined by the Delaunay triangulation will be studied through their aspect ratio. If the aspect ratio of a given tetrahedron results to be lower than a set value, this will be erased from the sample. A bar element B^{ij} will be then defined in each pair of connected nodes x^i and x^j, which will represent the forces between the two cells.

3.2. The vertex model

A set of connected vertices $\{y^I, \dots y^{N_{tet}}\}$ will be defining the cell boundaries in the sample. One vertex y^I will be associated to each of the tetrahedra $\mathcal{T}^I$. This way, each node i therefore, will be surrounded by a set of vertices $\{{y_i}^I, \dots, {y_i}^{N_i}\}$. Note that N_i is not a set number and therefore can result to be a different value in each node case and in each time-step.

The position of the vertices will be found through the barycentric tessellation. In this type of tessellation, the vertices of the network will be described as the barycenter found in each of the tetrahedra in the mesh. This type of tessellation will assure that the vertex will be always staying inside the tetrahedron, even when the Delaunay triangulation is deformed through the different phases of the process. This same feature couldn't be achieved by the Voronoi diagram, which makes use of the bisectors instead.

The position of vertex y^I is also defined by a local parametric coordinate ξ^I in each tetrahedron. Therefore, there will be as many local parametric coordinates $\{\xi^I, \dots, \xi^{N_{tet}}\}$ as tetrahedra in the mesh $\{\mathcal{T}^I, \dots, \mathcal{T}^{N_{tet}}\}$.

3.3. The hybrid model

This project will be carried through a hybrid model that will be formed by both the models presented previously. This hybrid model will provide the advantages from both the cell-centered and the vertex models: both the mechanical forces between cell-centers and the mechanical forces at the cell-cell junctions will be able to be studied at once. Also, the cell will be included as an essential unit, which will make the cell-cell contact transitions to be easier.

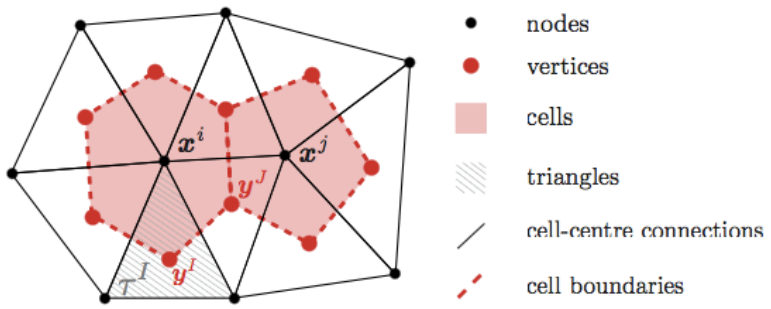


***Figure 3.1**. Two dimensional representation of the hybrid model. (Source: [3]).*

As shown in Figure 3.1, the hybrid model is the summation of the two others. In the figure, the cell-centered model is represented by the black elements, showing the nodes and the bar elements uniting them, which at the same time defines the Delaunay triangulation. The red elements, on the other hand, represent the vertex model, showing the different vertices around a set of nodes and the cell boundaries, which will define the cells area. An element obtained from the Delaunay triangulation is also shaded in grey in the figure.

4. Methodology

The process to be implemented in this study has several steps which need to be done in order to acquired the desired information from the program. These steps are summarized in Figure 4.1.

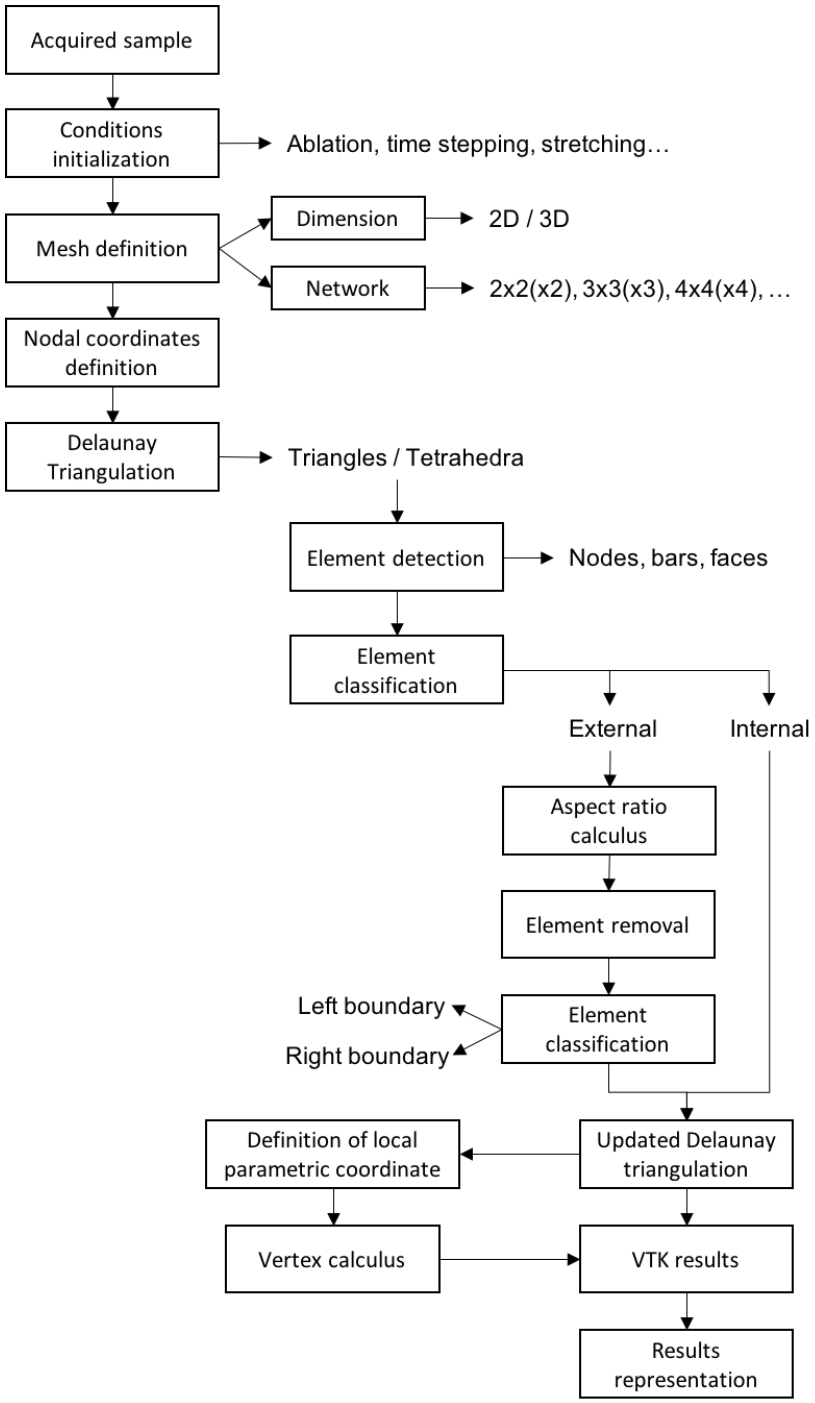


***Figure 4.1**. Scheme of the different steps that will take place during the process.*

4.1. The Main program

The whole process starts with the Main program, which defines the different variable values of the preliminary data, some of which are obtained in the experimental process and some other are to be set by the student running the program. These variables are crucial to the experiment because these are the ones to describe the process to be suffered by the sample during the process, or the total time of the experiment and the different time-steps to be performed.

In this program, the dimension of the sample and its network can be chosen. In the first of the two variables, only a two dimensional or a three dimensional approach can be chosen, while the network variable can take any dimension wished, as what this variable defines is the number of cells inside the sample.

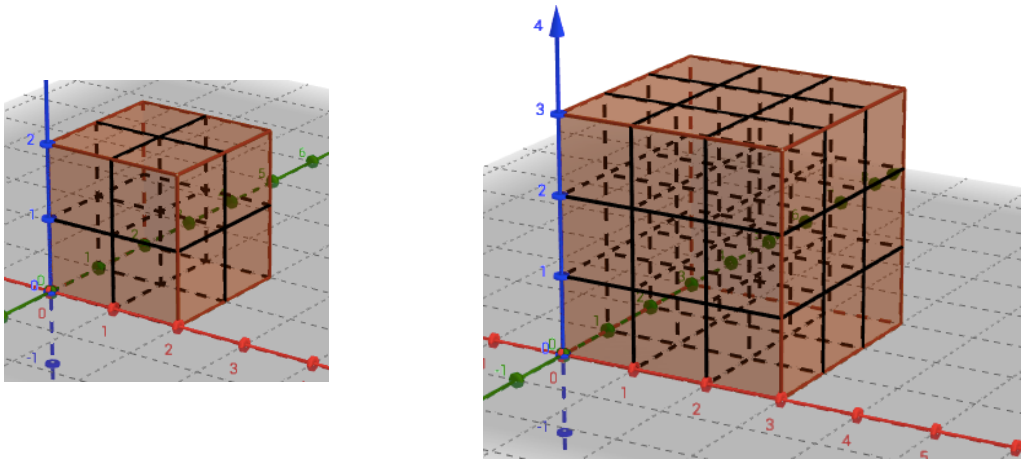


***Figure 4.2**. Representation of a 2x2x2 network (left) and a 3x3x3 network (right).*

These variables will also define the nodal Cartesian coordinates for the entire sample. Each nodal point will be located either on a vertex of the mesh or in an intersection between two of the divisor imaginary lines.

This program's code can be found in Annex A.1.

4.2. The Boundary Filter function

The Boundary Filter function is called in the Geo function and its aim is to filter external elements according to their aspect ratio. First of all, it creates the Delaunay triangulation according to the dimensional approach selected before. This triangulation introduces a variable which defines the bar

elements connecting two nodes of the triangles (2D) or the faces containing three nodes of the tetrahedra (3D).

These elements can be then classified into two different groups: the first of them is the one defining only the internal elements and the other is the one containing the external. This differentiation can be carried out by detecting which of the bar elements/face elements are being repeated and which are not: the ones appearing twice will be the ones sharing the element itself, and therefore, will be the ones considered as internal elements.

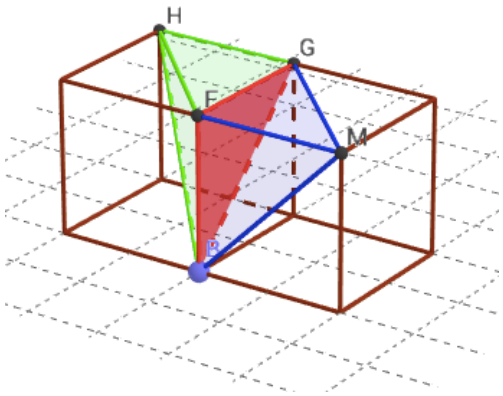


***Figure 4.3**. Example of internal and external elements in the mesh.*

In Figure 4.3. it can be seen how two different tetrahedra $\mathcal{T}^I$ and $\mathcal{T}^J$ share the face defined by the B – F – G nodes (x^b, x^f, x^g), and therefore happens to be an internal element. On the contrary, in this specific case where only two tetrahedra were drawn, all the other faces would be considered as external elements. This case is summarized in the Table 4.1.

The next thing this function does, is the calculation of the aspect ratio of each of the triangles/tetrahedra. In the 2D case, we make use of the ratio given by the division of the inradius by the circumradius to calculate the aspect ratio of each of the triangles. In case of this value being lower than a specific value established during the previous functions, this triangle will be erased from the sample.

Tetrahedron	Face	Nodes	Internal / External
1	**1**	x^b, x^f, x^g	Internal
1	**2**	x^f, x^g, x^m	External
1	**3**	x^b, x^g, x^m	External
1	**4**	x^b, x^f, x^m	External
2	**1**	x^b, x^f, x^g	Internal
2	**2**	x^f, x^g, x^h	External
2	**3**	x^b, x^g, x^h	External
2	**4**	x^b, x^f, x^h	External

***Table 4.1**. Classification of internal and external elements in Figure 4.2.*

In the three dimensional case, the aspect ratio of each of the tetrahedra defining the sample will be calculated through the ratio given between the maximum edge length and the minimum edge length found in each of the tetrahedra. As in the two dimensional approach, if this value is found to be lower than the one defined, the tetrahedron will be erased.

After this procedure, the program shows which are the external bars or faces, in 2D and 3D respectively, being these the ones to be on the boundary of the sample. This will be returned as an array containing the indices of the different nodal points creating each of these elements. Finally, the nodes of the network that have been left hanging will be detected and erased, in case of any.

Boundary Filter will return the network nodal points coordinates, the connectivity of each of the triangles/tetrahedra nodes, and another variable including only the bars connecting two nodes (in 2D) or the faces that contain three nodes (3D) which happen to be external in the network.

The Geo and the Boundary filter functions' codes can be found in Annex A.2 and Annex A.3, respectively.

4.2.1. Aspect Ratio Calculus

Although the Delaunay triangulation of the nodal network guarantees a maximum aspect ratio of the resulting triangles/tetrahedra when created, the network suffers a process of deformation, which will suppose also a deformation in the initial Delaunay triangulation due to mechanical equilibrium.

In order to keep this triangulation true, it will be set a certain tolerance, and whichever triangle or tetrahedron's aspect ratio is found to be lower, will be removed. This way, the process will make sure that the triangulation stays faithful to the initial one, not including new possible elements formed by the deformation process.

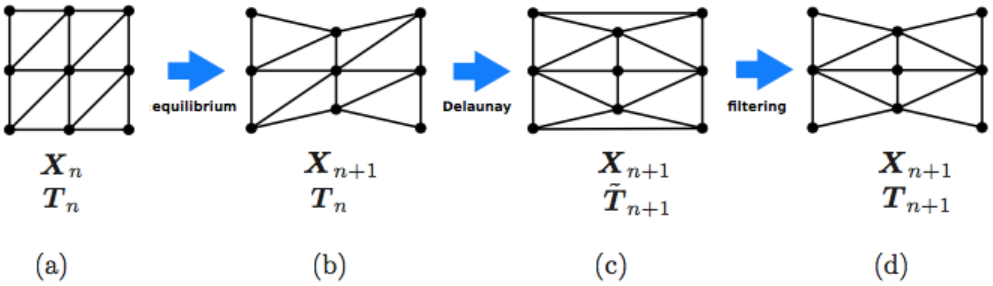


***Figure 4.4**. Schematic view of the filtering process of Delaunay triangulation. (Source: [3]).*

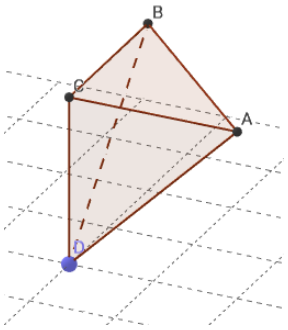
In Figure 4.4, step (a) represents the initial Delaunay triangulation in a two dimensional, 2x2 network (9 nodes). Once passed the deformation process, the triangulation turns out to be as shown in step (b), where the network has been deformed in order to keep mechanical equilibrium. Due to this change, the program will remodel the Delaunay triangulation, creating new connection bars and replacing those that already existed by those which happen to have a higher aspect ratio in the network (step (c)). This is when the filtering process needs to be done in order to eliminate those triangulation connections which are not meant to be in the sample. Through the calculation of the aspect ratio of each of the triangles and these being compared to a set tolerance, the program will be able to detect those elements which need to be removed from the sample. This process solution will be the one shown in step (d).

This process can be also used with tetrahedra in a three dimensional point of view. Therefore, it is necessary to calculate the aspect ratio of each of the elements in order to know which tetrahedra need to be removed and which don't.

The calculus of the aspect ratio in tetrahedra, as in triangles, can be done through two different approaches: through the inradius/circumradius ratio or through the ratio achieved by dividing the minimum-length-edge l by the maximum-length-edge L of the tetrahedron.

Inradius/Circumradius Ratio

In order to calculate the aspect ratio of a given tetrahedron through the inradius/circumradius ratio, it is necessary to know which four nodes are creating it, which will be designated as *A, B, C* and *D*. Each of these nodes will be defined by their own coordinates.



$$\begin{aligned} \boldsymbol{A} &= (A_x, A_y, A_z), \\ \boldsymbol{B} &= (B_x, B_y, B_z), \\ \boldsymbol{C} &= (C_x, C_y, C_z), \\ \boldsymbol{D} &= (D_x, D_y, D_z), \end{aligned} \qquad \textbf{(Eq. 4.1)}$$

***Figure 4.5**. Representation of a tetrahedron with its defining nodes.*

Then by calculating the following expressions [2],

$$\alpha = \begin{vmatrix} A_x & A_y & A_z & 1 \\ B_x & B_y & B_z & 1 \\ C_x & C_y & C_z & 1 \\ D_x & D_y & D_z & 1 \end{vmatrix}, \qquad \textbf{(Eq. 4.2)}$$

$$\begin{aligned} N_{ABC} &= (B - A) \times (C - A), \\ N_{ABD} &= (B - A) \times (D - A), \\ N_{ACD} &= (C - A) \times (D - A), \\ N_{BCD} &= (C - B) \times (D - B), \end{aligned} \qquad \textbf{(Eq. 4.3)}$$

$$\gamma = \begin{vmatrix} {A_x}^2 + {A_y}^2 + {A_z}^2 & A_x & A_y & A_z \\ {B_x}^2 + {B_y}^2 + {B_z}^2 & B_x & B_y & B_z \\ {C_x}^2 + {C_y}^2 + {C_z}^2 & C_x & C_y & C_z \\ {D_x}^2 + {D_y}^2 + {D_z}^2 & D_x & D_y & D_z \end{vmatrix}, \qquad \textbf{(Eq. 4.4)}$$

$$F_x = \begin{vmatrix} {A_x}^2 + {A_y}^2 + {A_z}^2 & A_y & A_z & 1 \\ {B_x}^2 + {B_y}^2 + {B_z}^2 & B_y & B_z & 1 \\ {C_x}^2 + {C_y}^2 + {C_z}^2 & C_y & C_z & 1 \\ {D_x}^2 + {D_y}^2 + {D_z}^2 & D_y & D_z & 1 \end{vmatrix}, \qquad \textbf{(Eq. 4.5)}$$

$$F_y = \begin{vmatrix} {A_x}^2 + {A_y}^2 + {A_z}^2 & A_x & A_z & 1 \\ {B_x}^2 + {B_y}^2 + {B_z}^2 & B_x & B_z & 1 \\ {C_x}^2 + {C_y}^2 + {C_z}^2 & C_x & C_z & 1 \\ {D_x}^2 + {D_y}^2 + {D_z}^2 & D_x & D_z & 1 \end{vmatrix}, \quad \textbf{(Eq. 4.6)}$$

$$F_z = \begin{vmatrix} {A_x}^2 + {A_y}^2 + {A_z}^2 & A_x & A_y & 1 \\ {B_x}^2 + {B_y}^2 + {B_z}^2 & B_x & B_y & 1 \\ {C_x}^2 + {C_y}^2 + {C_z}^2 & C_x & C_y & 1 \\ {D_x}^2 + {D_y}^2 + {D_z}^2 & D_x & D_y & 1 \end{vmatrix}, \quad \textbf{(Eq. 4.7)}$$

the inradius r, defined as the radius of the sphere tangential to the faces of the tetrahedron, can be calculated as

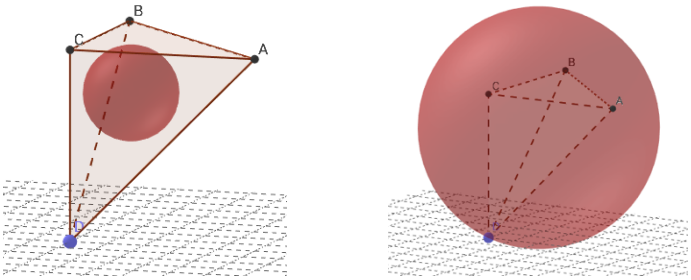
$$r = \frac{|\alpha|}{\|N_{ABC}\| + \|N_{ABD}\| + \|N_{ACD}\| + \|N_{BCD}\|}, \quad \textbf{(Eq. 4.8)}$$

whereas the circumradius R, defined as the radius of the sphere circumscribed to the tetrahedron, is computed as

$$R = \frac{\sqrt{{F_x}^2 + {F_y}^2 + {F_z}^2 + 4\alpha\gamma}}{2|\alpha|}, \quad \textbf{(Eq. 4.9)}$$

and therefore, the ratio w_{rR} can be given by the expression

$$w_{rR} = \frac{r}{R} = \frac{2|\alpha|^2}{(\|N_{ABC}\| + \|N_{ABD}\| + \|N_{ACD}\| + \|N_{BCD}\|)\sqrt{{F_x}^2 + {F_y}^2 + {F_z}^2 + 4\alpha\gamma}} \quad \textbf{(Eq. 4.10)}$$



(a) (b)

Figure 4.6. (a) *Inradius of a tetrahedron.*
(b) *Circumradius of a tetrahedron.*

Minimum length/maximum length ratio

This aspect ratio calculus makes use of the coordinates of the given tetrahedron as in the previous case (Eq. 4.1). Once these are known, the lengths of each of the six edges of each tetrahedron will be calculated the following way,

$$\begin{aligned} d_{AB} &= \|V_B - V_A\|, \\ d_{AC} &= \|V_C - V_A\|, \\ d_{AD} &= \|V_D - V_A\|, \\ d_{BC} &= \|V_C - V_B\|, \\ d_{BD} &= \|V_D - V_B\|, \\ d_{CD} &= \|V_D - V_C\|, \end{aligned} \qquad \textbf{(Eq. 4.11)}$$

this way, finding out which of the lengths is the biggest and which one is the smallest, being l the minimum length and L the maximum length, it will be possible to compute the ratio

$$w_{lL} = \frac{l}{L} \qquad \textbf{(Eq. 4.12)}$$

Comparison of the two ratios

This comparison can be calculated imagining one of the nodal points of the tetrahedron started to move away from the other three. This would make the spheres' radius and the lengths of the edges to change, resulting this into a variation of both aspect ratios values.

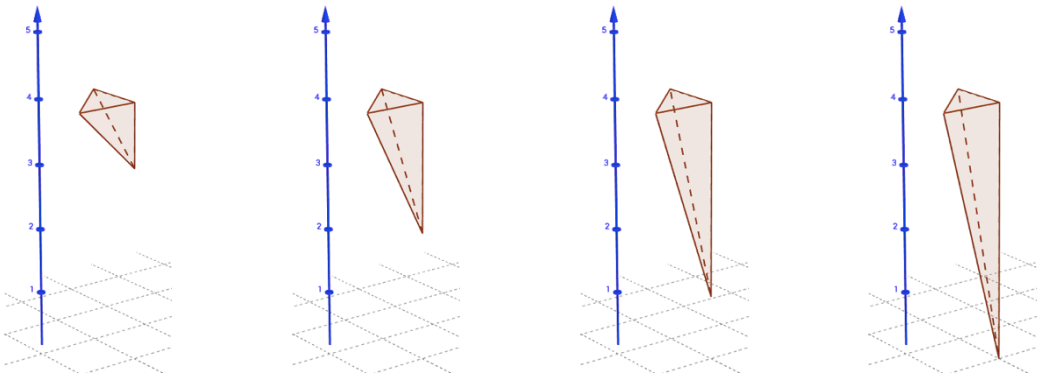
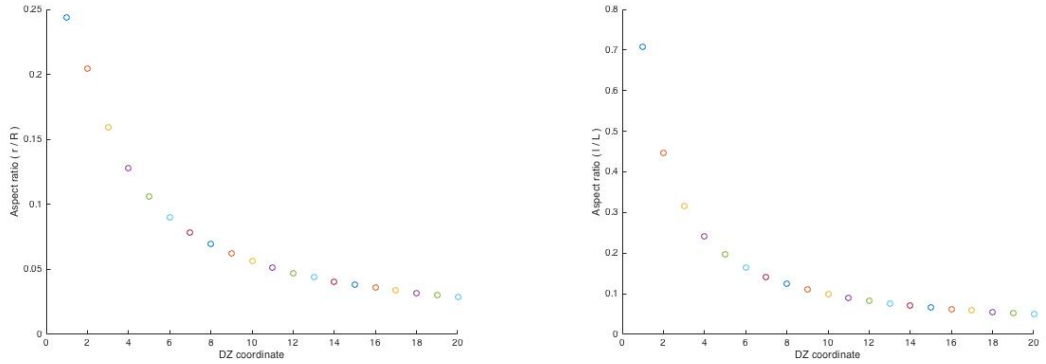


***Figure 4.7**. Evolution of a tetrahedron as one node moves away.*

In the case of the first mentioned ratio, it can be seen how the radius r of the sphere tangential to the faces of the tetrahedron will slightly vary, while the radius R of the sphere circumscribed to the tetrahedron will increase in each step of the process. Consequently, the aspect ratio w_{rR} will decrease in each step.

The same happens in the second mentioned ratio, where the minimum length l among the edges will be constant, whereas the maximum length L will increase, producing a decrease in the aspect ratio w_{lL} with each iteration.

If analyzed these cases, with a high number of iterations in both procedures, it is obtained that both ratios work the same way and provide good results as shown in the next figures,



(a) (b)

***Figure 4.8**. Comparison between both of the ratio calculus as distance increases.*
***(a)** r/R ratio**. (b)** l/L ratio.*

From these two figures, it can be extracted the threshold value for the aspect ratio. Even though both procedures are correct for this calculus, the one involving the lengths is easier to implement it and therefore, it doesn't introduce much option to errors as the one involving the inradius and the circumradius. Moreover, the difference between the aspect ratio values between the first iterations is higher than the one involving the radius ratio and therefore, the discriminative power of this ratio is higher than the other, which is a decisive point when choosing which ratio to use.

Figure 4.8 (b) shows how the aspect ratio decreases as the maximum length edge increases, therefore it could be implemented a threshold value around 0.35, where the distance turns to be approximately twice the minimum length edge. The tetrahedra that happen to have a higher aspect ratio than 0.35, will be erased from the sample, as it was not an original tetrahedron from the Delaunay triangulation.

4.3. The Set Left Right function

This function will come just right after Boundary Filter and will determine the left and the right boundaries of the sample. Each of the external elements defined previously will be processed in order to know which ones belong to the left boundary and which ones belong to the right. There will be also external elements that will not belong to neither of the boundaries but will belong in the top or the bottom boundaries.

First of all, a certain tolerance must be set. This tolerance will define which of the external elements are on the left boundary of the sample and which ones are on the right through the comparison with the angle of the normal of each of the elements. Therefore, this tolerance needs to be high enough so that this process can be reliable.

From a two dimensional point of view, two points will be defined and their nodal positions will be obtained. From these two points, the bar direction uniting them will be calculated and then rotated 90 degrees clockwise.

$$l_{1-2} = {x_1}^T - {x_2}^T \,, \qquad \textbf{(Eq. 4.13)}$$

$$Rotate_{90º} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} * \; l_{1-2} \,, \qquad \textbf{(Eq. 4.14)}$$

Then the normal to this bar it is obtained,

$$nT = \frac{Rotate_{90º}}{\|l_{1-2}\|} \qquad \textbf{(Eq. 4.15)}$$

If bars from the previous example are now seen as faces, the situation is switched over to a three dimensional point of view. The process though, will be carried out pretty much the same way. Each of these external faces, will be composed by three nodes. From these three, the program will calculate the two bars that connect one of the nodes to the other two. Then, it will calculate the cross product between these two bars and calculate its normal.

$$l_{1-2-3} = (x_2 - x_1) \times (x_3 - x_1), \quad \textbf{(Eq. 4.16)}$$

$$nT = \frac{l_{1-2-3}}{\|l_{1-2-3}\|} \quad \textbf{(Eq. 4.17)}$$

The following process works the same for both the two dimensional and the three dimensional points of view. Once calculated nT, its first coordinate (X coordinate) will be compared to the tolerance mentioned before. This first coordinate can achieve three different values ranges which need to be highlighted,

$$\begin{cases} nT > tolerance\ , \\ tolerance > nT > -tolerance \\ -tolerance > nT \end{cases} \quad \textbf{(Eq. 4.18)}$$

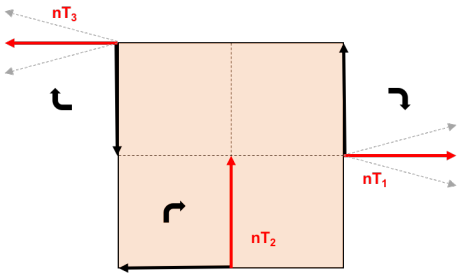


***Figure 4.9**. Representation of the possible cases in nT – tolerance comparison.*

where nT is the X coordinate for the normal of the bar/face element. It can be seen how in the first case in Eq. 4.18, which is represented as nT_1 in Figure 4.9, the external element is situated in the right boundary of the sample, which can be confirmed by nT_1, as its X coordinate results to be positive and higher than the tolerance. In case of nT_3, the element is located in the left boundary, as nT_3 X coordinate is negative and its absolute value is higher than the absolute value of the tolerance. In case of the nT_2 horizontal coordinate being in between the tolerances, the external element turns out to be either on the top or on the bottom boundary and, therefore, doesn't need to be classified in any of the previous two cases. The grey arrows represent the tolerances for each direction in the boundaries.

Set Left Right's code can be found in Annex A.4.

4.4. The Voronoi Interpol function

The construction of the vertex model starts with this function, where the different necessary data are initialized. It is going to be defined a new variable, designated as ξ^I that will define the local coordinate of vertex I in tetrahedron $\mathcal{T}^I$. In the initial time step, all the ξ^I values will be given a constant value $\xi^I = (1/4, 1/4, 1/4)$, which corresponds to the barycentric tessellation of the sample mesh in each of the tetrahedra.

Voronoi Initial then calls the Voronoi function, where the values of the vertex positions are processed in every time-step of the study. For the purpose of calculating a vertex position y^I it is necessary to calculate it through the local parametric coordinate ξ^I, belonging to triangle $\mathcal{T}^I$. In order to do so, it is used the interpolation

$$y^I = \sum_{i \epsilon \mathcal{T}^I} p^i(\xi^I) x^i \qquad \textbf{(Eq. 4.19)}$$

where the relation between the vertex position y^I and the four nodal positions in triangle $\mathcal{T}^I$ is given in concordance to the shape function $p^i(\xi^I)$. In Figure 4.10, it is represented an example in 2D on how the vertex coordinates are interpreted in each point of view, where ξ^I is 0.33, as it takes a two dimensional value.

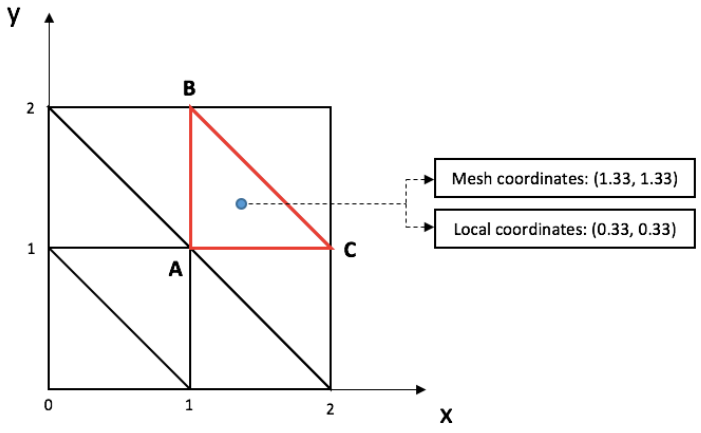


***Figure 4.10**. Representation of a vertex and its coordinates.*

This Voronoi Interpol and the Voronoi codes can be found in Annex A.5 and in Annex A.6, respectively.

4.5. The Output VTK function

Output VTK function creates a series of data according to the Visualization Toolkit (VTK) software format, which is used for 3D computer graphics, image processing and visualization. This data that can be used in a different program called Paraview. This program is able to represent this data and show the evolution of the sample through time, both from a two dimensional or three dimensional point of view.

Output VTK includes several other function, that are the ones to actually create the VTK data for the different models. In the VTK files acquired with these functions will be described the created mesh and the two different models: the cell-centered model and the vertex model. Once obtained the files and opened in Paraview, the student will be able to chose whether to show only one of the two models or to show them both at once.

Paraview will show as well a colored mesh that will be changing through the different time steps of the experiment. A color legend will be shown on the side to know which values on which variable are being represented through the different coloring. This feature will be of a high importance when studying the mechanical equilibrium in the cells.

4.5.1. VTK results comprehension

The VTK files need to be compiled in a certain way, so that the programs that can make use of these data can read them properly [5]. Therefore, in this section will be explained how the VTK files are synthesized and how to understand the data found in them.

The VTK files format is modeled by five basic parts.

1. **File version and identifier**: This part consists of a single line:

   ```
   #vtk DataFile Version x.x
   ```

   This line must be identical to the previously shown, except from the `x.x`, which stands for the version of the VTK release.

2. **Header**: This section can be used as a description of the data or to add any other pertinent information.

3. **File format**: The file format sets which type of file this is (ASCII or binary).

4. **Dataset structure**: Formed by a single line, this section describes the type of dataset. The line is composed by the word `DATASET` followed by a keyword which describes the type of dataset.

5. **Dataset attributes**: This part begins with a certain keyword followed by an integer number. This number will represent the number of data items in the dataset. This part must include `CELL_DATA` and `POINT_DATA`.

```
(1)  # vtk DataFile Version 3.98
(2)  Delaunay_vtk
(3)  ASCII
(4)  DATASET UNSTRUCTURED_GRID
(5)  POINTS 18 float
```

***Figure 4.11**. Parts of the VTK data file format.*

The previous figure has been extracted from a real case in this project. The line for the file version and the identifier is going to be the same for all of the obtained VTK files during the study. The header, will be different for each VTK file, as is the line where the data for each of them is described. Both the third and the fourth line are going to stay the same in each of the obtained files.

The dataset structure (4) is the specification which will define how the dataset attributes (5) are going to be organized. `UNSTRUCTURED_GRID` is chosen because this type of dataset structure consists of arbitrary combinations of any possible cell type. This structure is defined by `POINTS`, `CELLS` and `CELL_TYPES`. `POINTS` data defines the point coordinates. As for `CELLS`, it contains two parameters: the number of cells *n* and the the total number of integer values required to represent the list (*size*). `CELL_TYPES` section also contains *n*, and the data in it is an integer per cell that specifies the cell type.

1	Vertex	6	Triangle strip	11	Voxel
2	Poly vertex	7	Polygon	12	Hexahedron
3	Line	8	Pixel	13	Wedge
4	Poly line	9	Quadruple	14	Pyramid
5	Triangle	10	Tetrahedron		

***Table 4.2**. Linear cell types found in VTK.*

A real example on how to read the VTK files is going to be shown next, in Figure 4.12.

```
# vtk DataFile Version 3.98
Delaunay_vtk
ASCII
DATASET UNSTRUCTURED_GRID
POINTS 8 float
 0.333333 0.333333 0.000000
 0.333333 1.333333 0.000000
 0.666667 1.666667 0.000000
 1.333333 0.333333 0.000000
 1.666667 1.666667 0.000000
 1.666667 0.666667 0.000000
 0.666667 0.666667 0.000000
 1.333333 1.333333 0.000000
CELLS 10 34
3 5 7 2
3 5 2 1
3 5 1 6
3 5 6 3
2 1 2
2 1 6
2 2 7
2 3 5
2 3 6
2 5 7
CELL_TYPES 10
5
5
5
5
3
3
3
3
3
3
CELL_DATA 10
SCALARS stress float
LOOKUP_TABLE default
0.000000
0.000000
0.000000
0.000000
0.000000
0.000000
0.000000
0.000000
0.000000
0.000000
```

POINTS

- 8: Total number of possible nodal positions.
- float: High-precision fractional values for the data.
- First column: X coordinate.
- Second column: Y coordinate.
- Third column: Z coordinate.

CELLS

- 10: Number of cells of the sample.
- 34: Size of the cell list (total number of integers needed to represent the list).
- First column: Number of points creating the cell.
  - ⇒ 2: Line.
  - ⇒ 3: Triangle.
  - ⇒ 4: Tetrahedra.
- Rest of the columns: Index of the used points.

CELLS_TYPES

- 10: Number of cells of the sample.
- First column: Type of cell.
  - ⇒ 2: Point.
  - ⇒ 3: Line.
  - ⇒ 5: Triangle.
  - ⇒ 10: Tetrahedron.

CELLS_DATA

- 10: Number of cells of the sample.

***Figure 4.12**. Example of a resultant VTK file.*

5. Results

Once the Matlab code has ended, a sort of files will be obtained. Each of these will belong to one of the two following types: *VTKResultsxx.vtk* or *VTKResults.Vor.xx.vtk*, where *xx* stands for the time-step in the process of the VTK file. The first of these will contain the data of the cell-centered model while the other will contain the data for the vertex model approach.

When the resultant VTK files are obtained, these need to be exported to Paraview. These can be then plotted as seen in Figure 5.1, where both files have been selected to be plotted at the same time and in the same initial conditions.

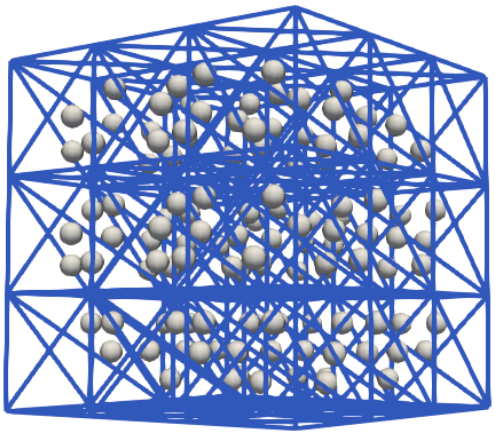


***Figure 5.1**. Representation of the results in a 3x3x3 network mesh, at the initial instant.*

In Figure 5.1 both the cell-centered model mesh and the vertex model mesh are represented. It can be seen how the edges in the representation are colored in blue and create the initial Delaunay triangulation. The white points that can be seen inside of these tetrahedra are the vertices found in the sample, found as the barycentric points in each of the tetrahedra.

A set of values for the total stress of each of the edges of the sample have been established and have been related to a color pattern. The coordinates of the nodes on one face of the cube have been fixed, experiencing a reaction force to the prescribed force exerted uniformly to the nodes on the other parallel face. As the process advances, the sample will evolve and so will the stresses and the positions of the edges. The vertex positions will vary, as the coordinates of the four nodes conforming the tetrahedron containing the vertex will also be modified. This will produce a different configuration in each of the time-steps of the process.

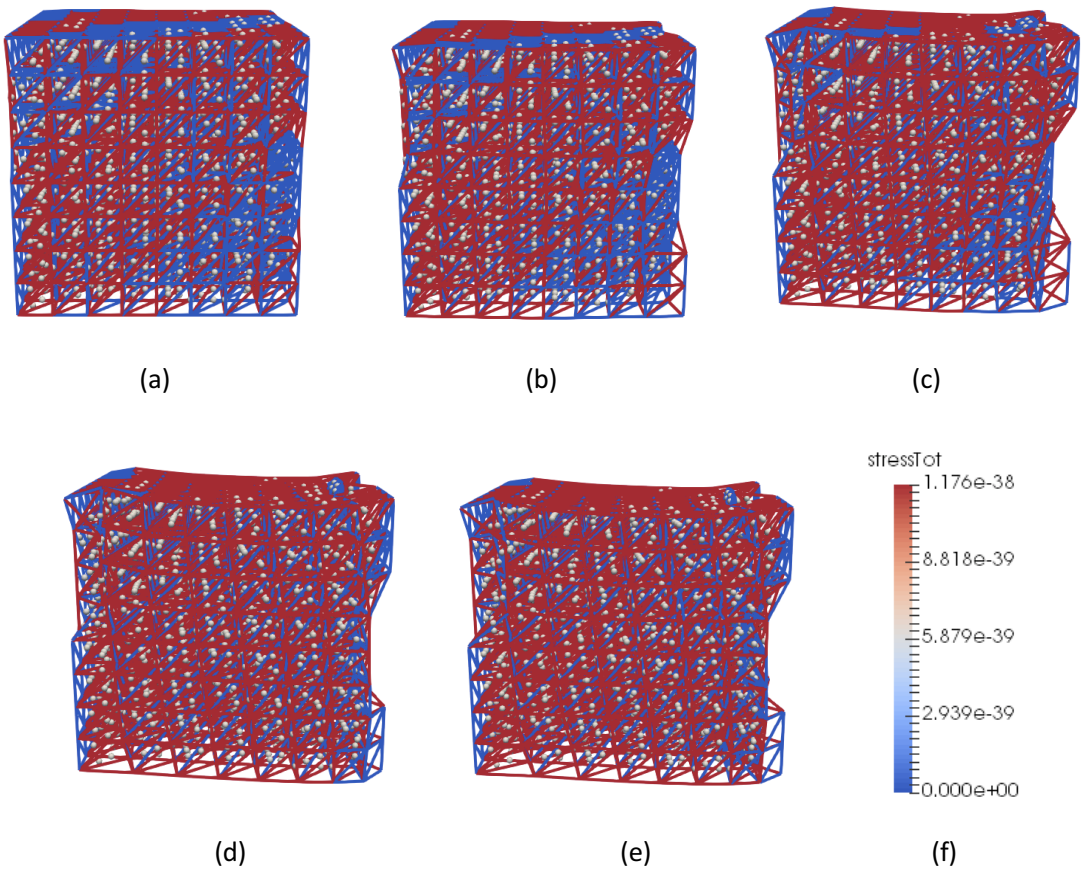


***Figure 5.2**. Sample geometry and stress evolution through time in an 8x8x8 network mesh.*
***(a)** 1st time-step, 0.05s. **(b)** 5th time-step, 0.25s. **(c)** 10th time-step, 0.50s. **(d)** 15th time-step, 0.75s.*
***(e)** Final instant, 1.00s. **(f)** Total stress color legend.*

The sample geometry, and therefore its mechanical equilibrium, changes as time passes, as it can be seen in Figure 5.2, from (a) to (e). Figure 5.2 (f) shows the color legend related to the stress found in each of the edges from the sample, which shows how, the closer to color red the edge is, the more stress there is in it. Therefore, it can be seen how the edges stress is increased in each time-step of the process, as in the first time-step (a), blue edges are found in the sample, but in the final instant, these are only found in the boundaries of the sample, as this is almost completely colored in red.

These figures also show how the sample gets shrunk and elongated at the same time. These two processes are perpendicular between them. The elongation of the sample is set to happen in the horizontal direction, which is the same direction as in which the force is performed. This will produce the shrinking to happen in the area perpendicular to the direction of the applied force.

Elongation is more pronounced in those edges found to be closer to the horizontal boundaries. Instead, the shrinking of the sample is more pronounced in the central section of the sample, where the maximum shrinking is found.

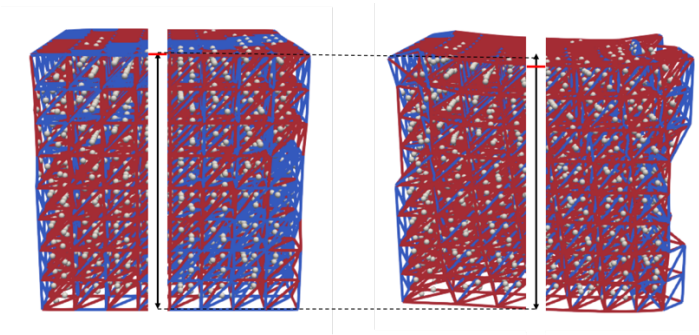


***Figure 5.3**. Illustration of the shrinking of the sample.*

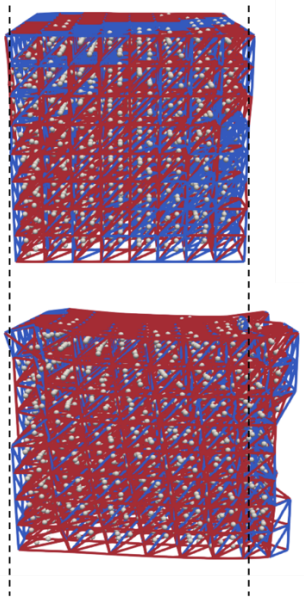


***Figure 5.4**. Illustration of the elongation of the sample.*

6. Budget

An economic evaluation of how much would it cost to carry out this project has been done considering there is the need to acquire everything needed. The economic evaluation can be divided in several types of expenses as it follows:

Hardware

Most of the project is developed in a computer, therefore it is necessary to acquire one and some peripherals which will help the student through the process.

Device	Price (€)
Computer	600,00
Peripherals	250,00
TOTAL	**850,00**

***Table 6.1**. Cost of the hardware required in the project.*

Software

The project requires the use of some programs in order to be carried out. These programs and their prices are shown in the next table.

Software	Price (€)
Matlab	2.000,00
Microsoft Office 2017	279,99
Paraview	0,00
TOTAL	**2.279,99**

***Table 6.2**. Cost of the software required in the project.*

Services

The service expenses consider energy consumption during the project realization, taking into account the fixed cost of energy and the energy cost of the devices used.

Device	Power (kW)	Time (h)	Cost (€/kW·h)	Total Cost (€)
Computer	0,25	960	0,10503	25,21
TOTAL				**25,21**

***Table 6.3**. Cost of the energy consumed by the devices.*

Personnel

It has been considered that in order to develop this project, a technician engineer is needed. Both the salary and the social security are considered for this worker.

$$Cost = Salary + Social\ Security \quad \textbf{(Eq. 6.1)}$$

According to the Spanish Social Security institution, the social security fee sits around the 32% and the 38% of the contribution basis, depending on the position held. Therefore, we can consider it to be approximately a third of the salary.

$$Social\ Security\ Fee = \frac{1}{3} * Salary \quad \textbf{(Eq. 6.2)}$$

If the workday is set to be eight hours a day, from Monday to Friday, and the employee is going to be hired for six months, considering each of the months to have four weeks, the salary will be calculated as it follows:

$$Salary = x\ \frac{€}{h} * 8\ \frac{h}{day} * 5\ \frac{days}{week} * 4\ \frac{weeks}{month} * 6\ months \quad \textbf{(Eq. 6.3)}$$

where *x* is the salary of the employee per hour, which in this case is set at 9.5 € per hour.

Personnel	Salary (€/6months)	Social Security Fee (€/6months)	Cost (€/6months)
Engineer	9.120,00	3.040,00	12.160,00
TOTAL			**12.160,00**

***Table 6.4**. Cost of the personnel required for the project.*

Total cost

The total cost of the project will be the sum of all the costs calculated above, which are summarized in the following table.

Expense Aspect	Cost (€)
Hardware	850,00
Software	2.279,99
Services	25,21
Personnel	12.160,00
TOTAL	**15.315,20**

***Table 6.5**. Summary of the project cost.*

7. Conclusions and future work

As shown in the previous section, the Matlab code was correctly developed in order to be able to study the evolution of the sample from a three dimensional point of view. From these results it can also be seen how a three dimensional approach is more exact than a two dimensional approach. This is due to the different actions that take place in all of the three spatial directions, and not only in two of them: although a two dimensional approach is a good way of studying complex three dimensional situations, it is never as realistic as a 3D study, as there are done many simplifications and considerations that introduce errors to the study. Instead, a three dimensional approach takes into account all the details of the evolving sample and therefore, provides more realistic results.

As said before, this is only a part of a much bigger study process and, therefore, there are many ways to continue this project in the future. The one that is most linked to this project proposed here, is to go one step further with this study with the vertex model approach. Once obtained the different vertices positions, these could be linked between them according to the adjacency of the tetrahedra containing them. This would provide a surface which contained all of the vertices sharing one same node in the sample. Once linked, these would create a new mesh in the sample which would represent the vertex model approach. With this, it could be possible to study also the variations suffered in the vertex model approach.

With this approach being studied together with the cell-centered model, the process could be studied in a much more reliable way. Moreover, when new variables were implemented (ablation, stress, …) the model would be much more realistic and true to reality.

8. Bibliography

8.1. Bibliographic References

[1] Ahrens, James, Geveci, Berk, Law, Charles (2005) *ParaView: An End-User Tool for Large Data Visualization*, Visualization Handbook, Elsevier, ISBN-13: 978-0123875822

[2] P. Mosaffa (2014) *Cell-Centre Model for Multi-Cellular Systems*. Thesis proposal.

[3] P. Mosaffa, A. Rodríguez-Ferran, J.J. Muñoz (2017) *Hybrid cell-centered/vertex model for the multicellular systems with equilibrium-preserving remodelling*. Wiley InterScience DOI: 10.1002/cnm.

[4] P. Mosasffa, Jose J. Muñoz, Yanlan Mao, Rob Tetley, Nina Asadipour, Antonio Rodríguez-Ferran (2016) *Cell-Centered/Vertex Model for Wound Healing*. Thesis proposal.

[5] VTK user's guide (2016) *VTK File Formats for VTK Version 4.2*. The VTK User's Guide.

8.2. Consulted Bibliography

[1] Conte V, Ulrich F, Baum B, Muñoz JJ, Veldhuis J, Brodland W, Miodownik M (2012) *A biomechanical analysis of ventral furrow formation in the Drosophila melanogaster embryo.* PLOS ONE 7(4):1–17.

[2] Francisco Bellot Rosado (2005) *Geometría del tetraedro*. Revista Escolar de la Olimpíada Iberoamericana de Matemática.

[3] Honda H, Tanemura M, Nagai T (2004) *A three-dimensional vertex dynamics cell model of space-filling polyhedra simulating cell behavior in a cell aggregate*. J Theor Biol 226:439–453.

[4] J Casey (1988) *A Sequel to the First Six Books of the Elements of Euclid, Containing an Easy Introduction to Modern Geometry with Numerous Examples*. Hodges, Figgis & Co., Dublin, 5th edition.

[5] Merks RMH, Glazier JA (2005) *A cell-centered approach to developmental biology*. Phys A 352(6):113–130.

[6] P. Mosaffa, N. Asadipour, D. Millán, A. Rodríguez-Ferran, J.J. Muñoz (2015) *Cell-centered model for the simulation of curved cellular monolayers*. Comp. Part. Mech., 2(4):359-370.

[7] Silvanus Alt, Poulami Ganguly, Guillaume Salbreux (2017) *Vertex models: from cell mechanics to tissue morphogenesis*. Phil. Trans. R. Soc. B 372: 20150520.

[8] Spahn P, Reuter R (2013) *A vertex model of Drosophila ventral furrow formation*. PLOS ONE 8(9):e75,051.

Annex A

A.1. The Main program

```
addpath(pwd)
clf;clc;
close all;
clearvars;

Dimension=2;     %Setting the dimension of the sample (2=2D; 3=3D)
Network=2;       %Setting the dimension of the sample ("Network x
                   Network" nodes per sample)

Mat.V.kappaA=0.1;              %Stiffness for Active/Maxwell 1st Branch on
                               Voronoi
Mat.D.kappa=1-Mat.V.kappaA;    %Stiffness for the 2nd branch Delaunay
Mat.V.GammaA=0.5               %Viscosity for Active 1st branch in
                               Voronoi, =0: no viscosity
Set.Vp=0.0;                    %Penalization of volume constraint (=0: no
                               area constraint)

Step.maxincr=5000;

%%%%%%%%%%%%% Boundary Condition %%%%%%%%%%%%%%%
BC.Monotonic=false;
BC.ts=0.0;
BC.xStrech    =0.3;  % Monotonic=false: Maximum load/displacement in
                        stretching
                     % Monotonic=true: stretching at t=BC.ts
BC.xStretch=[0 0.15];        % Monotonic=false: Maximum load/displacement
                               in stretching (% of total X length)
                              % Monotonic=true: stretching at t=BC.ts
BC.tStretch=[0 1];           % If tStretch defined, the stretch is defined
                           by the time hisotry (tStretch, xStretch), in
                           which case BC.Monotonic and BC.ts ae ignored.

%%%%%%%%%%%%%%%% Time Stepping %%%%%%%%%%%%%%%%%%
Step.dt0      =0.05; % Initial time-step
Step.tend     =1.0;

%%%%%%%%%%%%%%%% Ablation      %%%%%%%%%%%%%%%%%
Set.Ablation =0.0;      % =0: no tissue ablation, >0: tissue ablation
                          with redius "Ablation"(in cell mean size units)
Set.AbFullConstr=1; % =0: constrain left and right side of tissue only,
                    % =1: fully constrain triangular elements in contact
                    with external boundary
Ab.tAb=0.0;         % Time of applying Ablation
Ab.tEc=0;           % Wound ring contractility time lag (in units of
                        time-step)

%%%%%%%%%%%%%%%% Fitting settings %%%%%%%%%%%%%%%%%%
Set.File='ExpData_ECadCtrl_20160817_modelScale_addedPoints_addedContracti
      lity'; % File with data [gout, tout] that must be fitted
```

```
%%%%%%%%%%%%%%%%% Other settings %%%%%%%%%%%%%%%%%%%
Set.Dim=Dimension;
Set.OutputFile=0;   % =0 No outputfile
Set.print=1;        % =0 No printing
if ~exist('Ab','var')
    Ab.tAb=0.0;
end
[Ab,BC,err,esc,Mat,Set,Step]=PreSet(Ab,BC,Mat,Set,Step);
if err==1
    return;
end

%%%%%%%%%%%%%%%%% Geometry %%%%%%%%%%%%%%%%%%%
Set.Network =Network;   % Network >0: Geometry is made of a grid with
                          Network x Network
                       % Network =0: 1 element
                       % Network <0: Experimental real cells.
Set.ImageFile='handCorrection.png'; % File with cell images when
network<0

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%% Internal Setting BC %%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%% Internal Setting energy  %%%%%%%%%%%
Set.enerM=0;%=0: elstic force is k*eps^2 along bar directon
            %=1: elastic force is the derivative of the specific elastic
                  energy wrt x
               %(TRUE minimisation)
            %=2: elastic force is the derivative of the elemental elastic
                  energy wrt x
               %(TRUE minimisation, total energy takes into account
                  current length)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if ~exist('Mat','var')
    Mat.D.kappa=0;
    Mat.V.kappa=0;
end
if Set.Fitting
    % Define Material parameters to Fit:
    %  Delaunay:                                        Vertices:
    %  k kA kB GA GB eA eB Ec EcA EcB Ect EcAt b PE PV k kA kB GA GB eA
eB Ec EcA EcB Ect EcAt b
    p=[1 1  0  0  0  0  0  0  0   0   0   0    0  0 1 1 1  0  1  0  0  0
0  1   0   0   0    0  ];
    % Bounds
    LB=zeros(size(p));
    UB=Inf*ones(size(p));

[Mat,gout,u,r,Fval,exitflag,output,tout]=ActiveMaxwellFitting(Mat,Step,Se
t,BC,p,LB,UB,Set.File,Ab);
    savefile='gu';
    save(savefile,'Mat','r','Fval','exitflag','output','gout','tout');
    if Set.Ablation>1
        savefile2='V';
        save(savefile2,'V','Ab')
    end
else
```

```
    %profile on
    [gout,lr,tout,u,TL,ener,Ec,V,Ab]=ActiveMaxwell(Mat,Step,Set,BC,Ab);
    %profile viewer
    savefile='gu';
    save(savefile,'gout','tout','ener','Set','Mat');
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
PostSet(esc)
Subplots(gout,tout,u);
if Set.Ablation >0 && (Set.Network<0 || 2*Set.Ablation<Set.Network)
    num1=xlsread('area');
    num2=xlsread('APWound');
    Vplot(V,Ab,xlsread('area'),xlsread('APWound'));
end
disp('NetWork Ended')
```

A.2. The Geo function

```
function
[cdof,dim,lnodFE,dof,vext,nodes,TL,X,lnodExternal,nodAb]=Geo(BC,Set)

if Set.Dim==2
        (...)
elseif Set.Dim==3
    if Set.Network==0
        (…)
    elseif Set.Network>0    %3D network
        nx=Set.Network;     % Number of intervals along horizontal
                              direction (number of columns)
        ny=Set.Network;     % Number of intervals along vertical
                              direction (number of rows)
        nz=Set.Network;     % Number of intervals along diagonal
                              direction (number of layers)

        X=zeros((nx+1)*(ny+1)*(nz+1),3);
        K=0;
        for i=0:nx
            for j=0:ny
                for k=0:nz
                    K=K+1;
                    X(K,:)=[i j k];
                end
            end
        end
      [X,lnodFE,lnodExternal]=BoundaryFilter(X,Set.deltaR,Set.Bcells,Set.
      Network);
      [dof,cdof,vext] = SetLeftRight(Set.CVert,lnodExternal,Load,X);
      nodes=size(X,1);
      dim=size(X,2);
    else
        (...)
    end
end
nodAb=[];
end
```

A.3. The Boundary Filter Function

```
function [X,lnodFE,lnodExternal]=BoundaryFilter(X,deltaR,Bcells,Network)
 % Filters external elements with high aspect ratio
% Setting the triangle (2D) or tetrahedra (3D) connectivity
dim=size(X,2);
if dim==3 && max(X(:,3))-min(X(:,3))<eps
    dim=2;      %In case we have 3D but one of the dimensions is always
                  the same.
end
if dim==2
    lnodFE=delaunay(X(:,1),X(:,2)); %Triangles
elseif dim==3
    lnodFE=delaunay(X(:,1),X(:,2),X(:,3)); %Tetrahedra
end

% Filter elements with high aspect ratio
s=1;
while s>0
    F=zeros((dim+1)*size(lnodFE,1),dim+2);
    for i=1:size(lnodFE,1)
              F(i*(dim+1)-dim:i*(dim+1),:)=[nchoosek(lnodFE(i,:),dim)
              i*ones(dim+1,1) (1:dim+1)']; %%%matrix of triangles (edges
                  in 2-D) marked with the index of tetrahedron )
    end
    G=sort(F(:,1:dim),2);           %%%sorting each face ( edge in 2-D)
    H=[G F(:,dim+1:dim+2)];         %%%marking again triangles indices of
                                           original tetrahedron
    C=sortrows(H,1:dim);
    B=diff(C(:,1:dim));

    %%% finding list of repeated(in common)triangles(edges in 2-D)
    %defined by their index of original tetrahedron(triangle in 2-D)

    j=all(B==0,2); % find (internal) duplicated faces
    B_ind=[C(j,dim+1) C(j,dim+2)]; % List of internal faces: [element
                                     local_face_number]
    j=[0;j]; % B had one row less than C (returns double)
    j=logical(j); % turn j into logical (components)

    B_ind=[B_ind;C(j,dim+1) C(j,dim+2)]; % Store for the 2 elements the
                                           internal face belongs to
    lnod_ext=setdiff(F(:,dim+1:dim+2),B_ind,'rows');  %external faces
                  [global index of element local index of face]
    N_ext=unique(lnod_ext(:,1));
    lnodFE_ext=lnodFE(N_ext,:);  %defining external tetrahedra
    lnodFE(N_ext,:)=[];          %tetrahedra (triangles in 2-D) on
                  boundary are those which does not share all their faces

    s=0;
    for i=1:size(lnodFE_ext,1)   %%%filtering tetrahedrons with high
                                    aspect ration
```

```
        if dim==2
            (...)
        elseif dim==3
            va=lnodFE_ext(i,1);     %Detecting nodes of a tetrahedron
            vb=lnodFE_ext(i,2);
            vc=lnodFE_ext(i,3);
            vd=lnodFE_ext(i,4);

            VA=X(va,:);             %Detecting the position of each node
            VB=X(vb,:);
            VC=X(vc,:);
            VD=X(vd,:);

            alpha=det([VA 1; VB 1; VC 1; VD 1]);
            Nabc=cross((VB-VA),(VC-VA));
            Nabd=cross((VB-VA),(VD-VA));
            Nacd=cross((VC-VA),(VD-VA));
            Nbcd=cross((VC-VB),(VD-VB));

            %Calculus of the aspect ratio (w) through the l_min/l_max
            abc=norm(Nabc);
            abd=norm(Nabd);
            acd=norm(Nacd);
            bcd=norm(Nbcd);

            lengths=[abc; abd; acd; bcd];
            l_min=min(lengths);
            l_max=max(lengths);
            w=l_min/l_max;

%Calculus of the aspect ratio (w) through the inradius/circumradius
%gamma=det([sum(VA.^2,2) VA; sum(VB.^2,2) VB; sum(VC.^2,2) VC;
sum(VD.^2,2) VD]);
%FFX=det([sum(VA.^2,2) VA(2:3) 1; sum(VB.^2,2) VB(2:3) 1; sum(VC.^2,2)
VC(2:3) 1; sum(VD.^2,2) VD(2:3) 1]);
%FFY=det([sum(VA.^2,2) VA(1) VA(3) 1; sum(VB.^2,2) VB(1) VB(3) 1;
sum(VC.^2,2) VC(1) VC(3) 1; sum(VD.^2,2) VD(1) VD(3) 1]);
%FFZ=det([sum(VA.^2,2) VA(1:2) 1; sum(VB.^2,2) VB(1:2) 1; sum(VC.^2,2)
VC(1:2) 1; sum(VD.^2,2) VD(1:2) 1]);

%inradius=(abs(alpha))/(norm(Nabc)+norm(Nabd)+norm(Nacd)+norm(Nbcd));
%circumradius=(sqrt(FFX.^2+FFY.^2+FFZ.^2+4*alpha*gamma))/(2*abs(alpha));
%w=inradius/circumradius;

            if w<=deltaR
                s=s+1;
                lnodFE_ext(i,:)=0;
            end
        end
    end
    if s>0
        [e,~]=find(lnodFE_ext==0);
        lnodFE_ext(e,:)=[];
    end
    lnodFE=[lnodFE_ext;lnodFE]; % connectivty of external + internal
elements
end
```

```
%%% Find external bar elements on the boundary
lnodExternal=zeros(size(lnod_ext,1),dim);
for q=1:size(lnod_ext(:,1))

lnodExternal(q,:)=F(ismember(F(:,dim+1:dim+2),lnod_ext(q,:),'rows'),1:dim
);
    if lnod_ext(q,2)==2
        lnodExternal(q,[1,2])=lnodExternal(q,[2,1]);
    end
end

% Remove hanging nodes of X
n=size(X,1);
Xaux=1:n;
Xaux2=1:n;
k=0;
for i=1:n
    if isempty(find(lnodFE==i,1))
        Xaux(i-k:n-1-k)=Xaux(i+1-k:n-k); %Eliminates the position which
                                           wasn't found.
        Xaux2(i+1:n)=Xaux2(i+1:n)-1; %Duplicates the position which
                                           wasn't found.
        k=k+1;
    end
end
if k>0
    Xaux(n-k+1:n)=[];
    X=X(Xaux,:);
    if dim==2 % 2D
        (...)
    else %3D
        lnodFE=[Xaux2(lnodFE(:,1))' Xaux2(lnodFE(:,2))'
                  Xaux2(lnodFE(:,3))' Xaux2(lnodFE(:,4))'];
        lnodExternal=[Xaux2(lnodExternal(:,1))' Xaux2(lnodExternal(:,2))'
                        Xaux2(lnodExternal(:,3))'];
    end
end
```

A.4. The Set Left Right function

```
function [dof,cdof,vext] = SetLeftRight(CVert,lnodExternal,Load,X)
%SetLeftRight: determines left and right side, and applies boundary
%conditions
tolx=0.8; % Tolerance for angle of normal for detectin Right or Left
            boundary
nodes=size(X,1);
dim=size(X,2);
dof=1:dim*nodes; %Possible freedom degrees (all possible number of nodes)
x=reshape(X',dim*nodes,1)'; %X matrix into one single row
k=0;
ir=0;
il=ir;
cdof=zeros(1,dim*size(lnodExternal,1)); %only restricted nodes (external)
dofdeleteR=zeros(1,2*dim*size(lnodExternal,1));
dofdeleteL=dofdeleteR;
```

```
for n=1:size(lnodExternal,1)
    if dim==2
        (...)
    else        %%%3D
        dof1=(lnodExternal(n,1)-1)*dim+1:lnodExternal(n,1)*dim;
        dof2=(lnodExternal(n,2)-1)*dim+1:lnodExternal(n,2)*dim;
        dof3=(lnodExternal(n,3)-1)*dim+1:lnodExternal(n,3)*dim;
        x1=x(dof1);     %Point 1
        x2=x(dof2);     %Point 2
        x3=x(dof3);     %Point 3

        ncross=cross(x2-x1,x3-x1);
        nT=ncross/norm(ncross);

if nT(1)>tolx  %Left
            if CVert
                dofdeleteL(il+1)=dof1(2);
                dofdeleteL(il+2)=dof2(2);
                dofdeleteL(il+3)=dof3(2);
                dofdeleteL(il+4)=dof1(3);
                dofdeleteL(il+5)=dof2(3);
                dofdeleteL(il+6)=dof3(3);
                il=il+6;
            end
            dofdeleteL(il+1)=dof1(1);
            dofdeleteL(il+2)=dof2(1);
            dofdeleteL(il+3)=dof3(1);
            il=il+3;
        elseif nT(1)<-tolx  %Right
            cdof(k+1)=dof1(1);
            cdof(k+2)=dof2(1);
            cdof(k+3)=dof3(1);
            k=k+3;
            if CVert % No displacements along y and z
                dofdeleteR(ir+1)=dof1(2);
                dofdeleteR(ir+2)=dof2(2);
                dofdeleteR(ir+3)=dof3(2);
                dofdeleteR(ir+4)=dof1(3);
                dofdeleteR(ir+5)=dof2(3);
                dofdeleteR(ir+6)=dof3(3);
                ir=ir+6;
            end
            if Load==0 % Load applied on right side
                dofdeleteR(ir+1)=dof1(1);
                dofdeleteR(ir+2)=dof2(1);
                dofdeleteR(ir+3)=dof3(1);
                ir=ir+3;
            end
        end
    end
end
cdof=cdof(1:k);
dofdelete=[dofdeleteR(1:ir) dofdeleteL(1:il)];
cdof=unique(cdof);
if ~CVert
    [~,IR]=min(x(dofdeleteR+1));
    [~,IL]=min(x(dofdeleteL+1));
    dofdelete=[dofdelete dofdeleteR(IR)+1 dofdeleteL(IL)+1];
end
```

```
dof(dofdelete)=[];
vext=zeros(1,nodes*dim);
if Load==1
    vext(cdof)=1;
    cdof=[];
end
end
```

A.5. The Voronoi Interpol function

```
function [Vor,N,xi]=VoronoiInterpol(X,lnodFE,tess,xi,i)

nodes=size(X,1);
dim=size(X,2);
Vor=zeros(size(lnodFE,1),dim);
if i==0
    if tess
        xi=Vor;
        for i=1:size(lnodFE,1)  %%obtaining Voronoi vertices.
            s=1/(2*((X(lnodFE(i,2),1)-
X(lnodFE(i,1),1))*(X(lnodFE(i,3),2)-X(lnodFE(i,1),2))-(X(lnodFE(i,3),1)-
X(lnodFE(i,1),1))*(X(lnodFE(i,2),2)-X(lnodFE(i,1),2))));
            Vor(i,:)=s*[(X(lnodFE(i,3),2)-
X(lnodFE(i,1),2))*(norm(X(lnodFE(i,2),:))^2-norm(X(lnodFE(i,1),:))^2)-
(X(lnodFE(i,2),2)-X(lnodFE(i,1),2))*(norm(X(lnodFE(i,3),:))^2-
norm(X(lnodFE(i,1),:))^2) (X(lnodFE(i,2),1)-
X(lnodFE(i,1),1))*(norm(X(lnodFE(i,3),:))^2-norm(X(lnodFE(i,1),:))^2)-
(X(lnodFE(i,3),1)-X(lnodFE(i,1),1))*(norm(X(lnodFE(i,2),:))^2-
norm(X(lnodFE(i,1),:))^2)];
            %%obtaining Vronoi vertices in local coordinates respected to
triangles edges
            s=[X(lnodFE(i,3),2)-X(lnodFE(i,1),2) X(lnodFE(i,1),2)-
X(lnodFE(i,2),2)
                X(lnodFE(i,1),1)-X(lnodFE(i,3),1) X(lnodFE(i,2),1)-
X(lnodFE(i,1),1)];
            xi(i,:)=(Vor(i,:)-X(lnodFE(i,1),1:2))*(s/det(s));
        end
        N=[1-xi(:,1)-xi(:,2) xi(:,1) xi(:,2)];
    else
        N=ones(size(lnodFE,1),dim+1)/(dim+1); %Local parametric
                                               coordinate E
        xi=ones(size(lnodFE,1),dim)/3;
        Vor=Voronoi(lnodFE,N,X);
    end
    xi=reshape(xi',size(lnodFE,1)*size(X,2),1); %Reshaping xi in a matrix
                                               with just 1 column
else
    XI=reshape(xi,size(X,2),size(lnodFE,1))';
    if dim==2
        N=[1-XI(:,1)-XI(:,2) XI(:,1) XI(:,2)];
    else
        N=[1-XI(:,1)-XI(:,2)-XI(:,3) XI(:,1) XI(:,2) XI(:,3)];
    end
    Vor=Voronoi(lnodFE,N,X);
end
end
```

A.6. The Voronoi function

```
function [Vor]=Voronoi(FE,N,X)
nodes=size(X,1);
dim=size(X,2);
Vor=zeros(size(FE,1),dim);

for i=1:size(FE,1)
    if dim==2

Vor(i,:)=N(i,1)*X(FE(i,1),:)+N(i,2)*X(FE(i,2),:)+N(i,3)*X(FE(i,3),:);
    elseif dim==3

Vor(i,:)=N(i,1)*X(FE(i,1),:)+N(i,2)*X(FE(i,2),:)+N(i,3)*X(FE(i,3),:)+N(i,
4)*X(FE(i,4),:);
    end
end
end
```