Francisco José Benavides Murillo

MÉTODOS COMPUTACIONAIS PARA A ESTIMATIVA DOS COEFICIENTES ELASTICOS A PARTIR DE IMAGENS TOMOGRÁFICAS DE AMOSTRAS DE ROCHAS

INGLÊS

Dissertação apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Mestre. Área de Concentração: Computação Científica e Sistemas de Potência.

Orientador: Prof. Dr. Ricardo Leiderman

Niterói

2014

FRANCISCO JOSÉ BENAVIDES MURILLO

MÉTODOS COMPUTACIONAIS PARA A ESTIMATIVA DOS COEFICIENTES ELASTICOS A PARTIR DE IMAGENS TOMOGRÁFICAS DE AMOSTRAS DE ROCHAS

Dissertação apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Mestre. Área de Concentração: Computação Científica e Sistemas de Potência.

Aprovada em Agosto de 2014.

BANCA EXAMINADORA

_____

Prof. Dr. Ricardo Leiderman – Orientador

UFF

_____

Prof. Dr. Marcos Lage

UFF

_____

Prof. Dr. Daniel Castello

Universidade Federal de Rio de Janeiro

Niterói

2014

A mi madre

COMPUTER METHODS TO ESTIMATE ELASTIC COEFFICIENTS USING TOMOGRAPHIC IMAGES OF ROCK SAMPLES

GREETINGS

BRITISH GAS, SPONSOR OF THE PETROPHYSICAL PROJECT RESEARCH AT UFF

DIGITAL PETROPHYSICS

ABSTRACT

We derive estimates of the elastic properties of rock samples using their tomographic images. These estimates are achieved through a numerical simulation of the elastostatic equation, solving it with the finite element method and applying boundary conditions associated to the uniaxial compression test. The simulation software is developed and described. It uses CPU parallelization techniques, and is object oriented. The used computer language is C++, but we focus on the main algorithms description and computational decisions. The software is validated using exact analytical equations and approximate models. A real rock sample is used to analyze porosity and pore geometry influence on the elastic properties. The rock sample level of isotropy is also quantized. This still can be considered a work in progress, and the future research paths and possible consequences of these analyses are described along the text.

Keywords: Rock physics, Petrophysics, C++, Microtomography, Images, Numerical simulation

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

DRP - Digital Rock Physics

FEM - Finite element method

RP - Rock Physics

DIP - Digital image processing

OpenMP - Open Multi Processing

SIMD - Single Instruction, multiple data

TABLE OF CONTENTS

INTRODUCTION

Rock physics can be defined as the study of rock properties and their interactions with fluids. Petrophysics is the application of Rock Physics to Hydrocarbon Industry [12,39]. The geologic material inside a petroleum reservoir contains a three dimension network of interconnected pores through which a fluid flows. Knowledge of the physical properties associated to this pore network leads to the efficient design of the injection-production system of the reservoir, for economy of energy and maximization of hydrocarbon production.

Digital Rock Physics (DRP) is an interdisciplinary area that consists in imaging and digitizing the pore space and mineral matrix of a natural rock and then, numerically, simulating its physical processes. These numerical experiments can describe several types of rock macroscopic properties such as permeability, electrical conductivity and elastic module [3]. DRP is still an emerging technology which adds a numerical approach to the study of a reservoir structure and allows a better understanding to correlations between pore microstructure and physical properties of rocks [28].

This work scope is limited to elastic properties of rocks. These properties are used to drilling design, well completion and field development programs in the oil reservoir [39]. In general, these mechanical properties are acquired from laboratory experiments, or are estimated with oversimplified approximate models [3]. As we can expect, the laboratory experiments provide richer and more useful information which can also be compared to real geological phenomena, despite the fact of being executed on a totally different scale and time length [12]. This comparison makes accurate relationships between pore structure and elastic properties of rocks a fundamental and long standing problem in geophysics [5].

Therefore, laboratory experiments to obtain elastic properties of rocks have direct applications in geology and oil industry. The possibility to perform these tests in a computer simulated environment has the following advantages:

- Computer tests are cheaper;
- It is possible to create mechanical tests with the same rock sample and different conditions. This cannot be reproduced in the physical world;

- Digital analysis does not require big rock samples. The data can be damaged and can be obtained from reservoir areas in which a traditional laboratory analysis is not possible.

We can identify three steps in Digital Rock Physics:

- Image acquisition and segmentation;
- Numerical simulation;
- Interpretation of results.

In the next chapters we consider each one of these steps. Image acquisition is assumed, using Martin Blunt's examples from Imperial College [7]. These examples can be downloaded from the university web site.

Recently, the “Universidade Federal Fluminense”, sponsored by the British Gas Company [23], acquired tomographic equipment with which it is possible to image real samples of rocks. These images can be used as input data to rock physics computer software that simulates its physical properties. Our present work represents a starting point in this objective, describing the details of an implementation of a computer program that calculates the elastic rock properties. Even though there are other implementations [14], we made a new cross-platform approach in C++ language. We strongly believe that this leads to a better understanding of these technologies development.

The numerical simulations solve the mechanical based differential equation known as elastostatic equation. It is solved using the method of finite elements. In Chapter 1 we describe the numerical and mechanical background for this modeling. Chapter 2 is the most technical of all, describing the implementation details. The chosen language is C++, with parallel technologies such as OpenMP and modern CPU vectorization [38]. We do not describe language issues, and try to focalize in the algorithms using high level considerations. In Chapter 3 we become users of the developed software, executing experiments with synthetic and real rock sample images.

CHAPTER 1. BASIC CONCEPTS

1.1 BIBLIOGRAPHIC REVIEW

The different steps in digital petrophysics workflow are described in [4,3]. In general, the first step is to process the microtomographic images with basic digital image processing techniques. Sometimes this processing is limited to distinguish pore from rock material (biphasic samples). After this processing, a three dimension model of the rock sample is produced. This model is suitable to a numerical method application, such as finite element method (for mechanic properties), finite difference methods (for electrical properties) and other discrete methods. As we focus on elastic properties, our workflow handles 3D mesh models suitable for the finite element method.

The current state of the art in computer memory and speed makes it possible to handle the large set of data that a 3D image based finite element model contains [35]. To obtain elastic properties, the preferred numerical tool is the Galerkin finite element method [24]. This is the best method to solve the associated elastostatic equation as it minimizes the displacement energy potential [29,34].

The set of computational tools that are necessary for this numerical approach has been developed by Garboczi in Fortran language and it has been used for rock sample image resolutions that can be evaluated in common workstations [14,35]. As far as we know, larger distributed computer systems have not been used to solve these models.

Reitbergen et al. [42] describe the mathematical techniques and memory representation of large regular finite element meshes. They use the classical “element-by-element” finite element method [24], which suits very well to these kinds of data structures. Their range of applications includes medical, imaging human bones to obtain their elastic module.

For more than fifty years, a lot of research has been produced to obtain relationships between rock porosity (ratio between pore and material volume) and elastic properties.

This research makes an estimation of the elastic module based on simplistic pore geometries, providing precise lower and upper boundaries [18,20,19] or only approximations [32,11,25]. Despite the simplified assumptions these analyses make, they provide approximations with relative errors of less than 20% [9]. As we will show later, these simple pore geometries are very easy to reproduce in a computer generated model, providing an adequate mechanism to validate our computational results.

In this work we are focused in reproducing real laboratory rock physics experiments in a computer simulation. One of the simplest test is the unconfined compression test [30], designed to get the elastic module of a rock sample [37,35]. There are also other tests, suitable for similar numerical techniques [27].

The random placement of pores can be captured by the micro-tomographic images and has a direct influence on the sample elastic module. If there is enough number of pores inside the sample, the laboratory unconfined test will give an adequate estimation of the media global module and it will not be necessary to consider periodic boundary conditions, in which the numerical method is applied as if the rock sample were a repeated pattern. Despite they cannot be reproduced in real laboratory tests, in most cases, periodic conditions are assumed to improve the estimation [5]. This work does not apply periodic boundary conditions, as it cannot be reproduced in the laboratory. In a future work, a comparison between using or not periodic conditions will be performed.

It is a well proven fact that all rock physical properties can be obtained from the microtomographic images [5]. The main focus in this work is three dimension analyses for which there are very few analytical exact expressions. One of them is given by Hashin [18], for a dilute concentration of spherical inclusions inside a matrix. Hashin also deduces the most precise upper and lower limits for the Young module [19,20,21]. Nevertheless it is almost impossible to find exact analytical expressions for the Young modulus for different porosities and arbitrary pore geometry and distributions [22]. This is the reason for which the numerical simulation is the only reasonable alternative.

1.2 IMAGE ACQUISITION

To simulate our physical experiment, we require a sequence of images of the rock sample. These images must be large enough to capture the different rock pore geometries. The micro-scale x-ray computed tomography enables the measurement of the local x-ray absorption in a small, rock sample, with a typical volume of a few cube millimeters or less. The resolution can be as small as 3 μm [7]. The obtained radiographs are used to create a sequence of gray scale images, in which the brightness is proportional to the x-ray level of absorption of the pixel [31]. We can identify these brightness levels with different kinds of rock materials. In other words, each pixel color is used to tag and identify mineral phases within the sample. In many cases we consider only two phases: pore and material. This is the biphasic dry sample in which the three dimensional model of the sample contains only one set of mechanical characteristics. It can be visualized as a simple 3D image made of binary voxels (Figure 1). The resolution of these images can be of 2000^3 voxels in some devices [7].

It is obvious that in order to obtain a numerical model, it is not enough to get the microtomograph output, but the images must be processed in order to suppress noise and ring artifacts using 3D digital image processing techniques: morphology, cluster analysis and segmentation algorithms [3]. The method with which the mesh is created is described in Chapter 2.

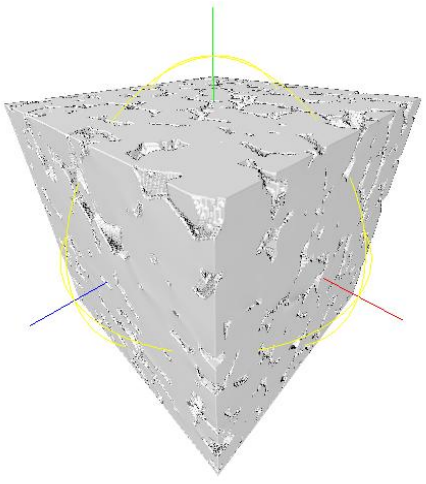

Figure 1. Three dimensional model of a rock sample. This three dimension model is generated with a PLY mesh file whose faces are the border faces of the border elements.

In this work we consider this sequence of images as an input to our workflow. This input can be viewed as a unique 3D image made of voxels, or as a sequence of 2D images made of pixels.

1.3 MATHEMATICAL MODEL

1.3.1 HOOKE'S LAW

In the theory of solid mechanics, bodies are considered deformable. This deformation depends on their internal forces which are modeled by stresses. The average stress on a cross-section area ΔA is defined as the ratio $\Delta \mathrm{F}/\Delta A$ where $\Delta \mathrm{F}$ is the force applied on this area [34]. If $\Delta A = \Delta A_x$ is a cross-section area perpendicular to X-axis, then the three components of stress are defined as:

$$\sigma_{xx} = \lim_{\Delta A_x \to 0} \frac{\Delta \mathrm{F_x}}{\Delta A_x} \qquad \sigma_{xy} = \lim_{\Delta A_x \to 0} \frac{\Delta \mathrm{F_y}}{\Delta A_x}$$

$$\sigma_{xz} = \lim_{\Delta A_x \to 0} \frac{\Delta \mathrm{F_z}}{\Delta A_x}$$

Where the first sub index denotes the direction of the force, and the second denotes the normal vector of the surface where it is being applied. There are three possible normal vectors in a three dimensional system, and at the same time, three different force directions.

Therefore, these combined stresses cannot be represented by a single vector. It is represented by a 3×3 matrix in which the first subindex denotes the number of the axis, and the second subindex, the normal vector of the surface (Figure 2):

$$\sigma = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{pmatrix} = \begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{pmatrix}$$

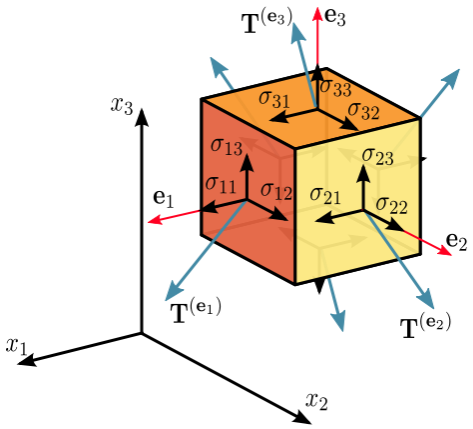

Figure 2. Stress tensor diagram. The motion equation is obtained supposing linearity inside this infinitesimal size box [37].

Let us suppose that u, v, w are the displacement fields of the differential element as functions of x, y, z in each axis. Then, these displacements can be written linearly as:

$$du = \frac{\partial u}{\partial x}dx + \frac{du}{dy}dy + \frac{du}{dz}dz$$
$$dv = \frac{\partial v}{\partial x}dx + \frac{dv}{dy}dy + \frac{dv}{dz}dz$$
$$dw = \frac{\partial w}{\partial x}dx + \frac{dw}{dy}dy + \frac{dw}{dz}dz$$

Or as:

$$\begin{pmatrix} du \\ dv \\ dw \end{pmatrix} = \begin{pmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \\ \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z} \end{pmatrix} \begin{pmatrix} dx \\ dy \\ dz \end{pmatrix}$$

The matrix of partial derivatives can be decomposed in its symmetric and antisymmetric parts:

$$\begin{pmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \\ \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z} \end{pmatrix} = \begin{pmatrix} \frac{\partial u}{\partial x} & \frac{1}{2}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) & \frac{1}{2}\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) \\ \frac{1}{2}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) & \frac{\partial v}{\partial y} & \frac{1}{2}\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right) \\ \frac{1}{2}\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) & \frac{1}{2}\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right) & \frac{\partial w}{\partial z} \end{pmatrix}$$

$$+\begin{pmatrix} 0 & \frac{1}{2}\left(\frac{\partial u}{\partial y}-\frac{\partial v}{\partial x}\right) & \frac{1}{2}\left(\frac{\partial u}{\partial z}-\frac{\partial w}{\partial x}\right) \\ -\frac{1}{2}\left(\frac{\partial u}{\partial y}-\frac{\partial v}{\partial x}\right) & 0 & \frac{1}{2}\left(\frac{\partial v}{\partial z}-\frac{\partial w}{\partial y}\right) \\ -\frac{1}{2}\left(\frac{\partial u}{\partial z}-\frac{\partial w}{\partial x}\right) & -\frac{1}{2}\left(\frac{\partial v}{\partial z}-\frac{\partial w}{\partial y}\right) & 0 \end{pmatrix}$$

The symmetric part is called "Symmetric strain tensor" [37] and the antisymmetric part, the "Rotation tensor" [24]. The symmetric tensor entries are denoted by ϵ_{ij} ($1 \leq i \leq 3$, $1 \leq j \leq 3$).

From now, we will use index notation in which double repeated index implies summation, and differentiation is denoted by a comma. When an index is not repeated, it denotes vector entries [24]. We define $u_i = (u_1, u_2, u_3)$. Then, the symmetric strain tensor can be described by the following abbreviated equation:

$$\epsilon_{ij} = u_{(i,j)} = \frac{1}{2}(u_{i,j} + u_{j,i}) \tag{1.3.1}$$

Hooke´s law establishes a relationship between the symmetric strain tensor and the symmetric stress tensor. In index notation it is written as:

$$\sigma_{ij} = c_{ijkl}\epsilon_{kl} \tag{1.3.2}$$

Where the coefficients c_{ijkl} constitute what is called the stiffness tensor. It is important to check the units of these equations.

The values of ϵ_{ij} do not have units, because it is a division between two lengths. As stress tensor has units N/m^2 (Newton per square meter) in the international system then the constants c_{ijkl} units are N/m^2 too. In Voigt notation, Hooke´s law is written avoiding repetition of symmetric components, in the following way [30]:

$$T = \begin{pmatrix} \sigma_1 = \sigma_{11} \\ \sigma_2 = \sigma_{22} \\ \sigma_3 = \sigma_{33} \\ \sigma_4 = \sigma_{23} \\ \sigma_5 = \sigma_{13} \\ \sigma_6 = \sigma_{12} \end{pmatrix} \qquad D_E = \begin{pmatrix} \epsilon_1 = \epsilon_{11} \\ \epsilon_2 = \epsilon_{22} \\ \epsilon_3 = \epsilon_{33} \\ \epsilon_4 = \epsilon_{23} \\ \epsilon_5 = \epsilon_{13} \\ \epsilon_6 = \epsilon_{12} \end{pmatrix}$$

Hooke´s law can be simplified, considering an isotropic medium, in which only two constants are necessary to specify stress-strain relation. Taking this into account, we get the simpler relation:

$$T = \begin{pmatrix} 2\mu+\lambda & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & 2\mu+\lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & 2\mu+\lambda & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{pmatrix} D_E = K_E D_E \tag{1.3.3}$$

The 6×6 matrix K is a simpler way to represent the stiffness tensor. The constants μ and λ are called Lamé constants, μ is the shear modulus or Lamé´s second parameter and λ is the Lamé first parameter [37]. All these constants have international unit: N/m^2. Under these new assumptions, Hooke´s law takes the following form:

$$\sigma_{ij} = \lambda \delta_{ij} \epsilon_{\alpha\alpha} + 2\mu \epsilon_{ij} \tag{1.3.4}$$

This is called the isotropic form of Hooke's law [30]. Here $\delta_{ij} = 0 \Leftrightarrow i \neq j$, $\delta_{jj} = 1 \forall j = 1,2,3$ and $\epsilon_{\alpha\alpha}$ is a sum over repeated index.

1.3.2 LINEAR ELASTOSTATIC EQUATION

To numerically reproduce our experimental test, the differential elastostatic equation must be solved. This equation models the material as continua. In each point of the domain (the three dimension rock sample) the second law of Newton must be accomplished. In the context of the elastostatic equation, this means that the addition of all forces must be zero. Using index notation, we have [24]:

$$\sigma_{ij,j} + f_i = 0$$

Where f_i is an external stress. This equation is deduced by using linearity in each infinitesimal domain element [37]. In our case, there are no sources of stress in any point of the domain. This implies that $f_i = 0$. The equation, then, becomes:

$$\sigma_{ij,j} = 0 \tag{1.3.5}$$

This is called the motion equation. All stresses are unknown variables, and this equation represents a system of 3 equations. Therefore, here we have 3 equations and 6 unknown variables (stress tensor is symmetric).

Now, we consider boundary conditions. They are important to produce more equations and make the system solvable. In our experiment, boundary conditions have an physical interpretation:

- Boundary points in which the displacement is known. These are the points in the rock samples that make contact with the stress device, that is, the lower and upper part of the rock. These points will be denoted by T_G.
- Boundary points in which the stress is known. These are the boundary points in which no stress is applied. These points will be denoted by T_H.

So, we have a new set of equations:

$$u_i = q_i \quad on \;\; T_G \tag{1.3.6}$$

Where q_i is a boundary displacement. We also have:

$$\sigma_{ij} n_j = 0 \quad on \;\; T_H \tag{1.3.7}$$

Where n_j are the entries of the normal vector of the surface. These equations produce a new set of 6 equations. But we have added three unknown variables: the displacement vector entries u_i. All these equations represent a set of 9 equations with 9 unknown variables. This means that the system is solvable, but this only happens in the domain boundary.

We are going to analyze what happens in the domain inner points. Hooke´s law creates a set of 6 more equations (because of the symmetry).

$$\sigma_{ij} = c_{ijkl}\epsilon_{kl} \tag{1.3.8}$$

But at the same time, the displacement tensor adds 6 unknown variables. The relation:

$$\epsilon_{ij} = \frac{1}{2}\left(u_{i,j} + u_{j,i}\right) \tag{1.3.9}$$

Creates 6 more equations and 3 unknown variables. At the inner points of the domain we have 15 equations and 15 unknown variables, so, the system is solvable. These systems of equations are globally called the elastostatic linear equation. The solution of this system in each domain point provides the displacement vector field, the displacement tensors and the stress symmetric tensor. It is supposed that all the functions involved are continuous and differentiable.

1.3.3 FINITE ELEMENT METHOD

It is obvious that solving the continuous elastostatic equation for each domain point is not computationally feasible. To make the problem, computationally tractable, it must be completely discretized. A level of discretization is achieved by supposing that the solutions can be approximated by linear combinations of simpler functions. We take the displacement vector field u_i and approximate it by $\tilde{u}_i$:

$$\tilde{u}_i(x,y,z) = \sum_{A=1}^{N} c_{Ai}N_A(x,y,z) + g_i(x,y,z) \qquad i = 1,2,3 \tag{1.3.10}$$

Without adding on index i. The functions N_{Ai} are simpler scalar functions that have numerical desirable properties which will be described later. They are called shape functions. The function g_i satisfies the boundary conditions in T_G.

This approximation can be visualized as an orthographic projection in which the function u_i is approximated in the space of functions S generated by $\{N_{Ai}\}_{A=1,..,N}$ and g_i [29].

In this set of generated functions, there's only one that best approximates u_i [29]. This best approximation has a geometrical interpretation. As we show in Figure 3, in finite dimension inner product vector spaces, the best approximation for u_i is given by the orthogonal projection, defined by the constants c_{iA} as shown in Equation (1.3.10). In our case, the space dimension of S is N (the number of shape functions) and u_i is not contained in it. To get the best approximation of u_i in S, we only have to find $3N$ scalars c_{iA}. This is the geometric idea that sustains the finite element method [29].

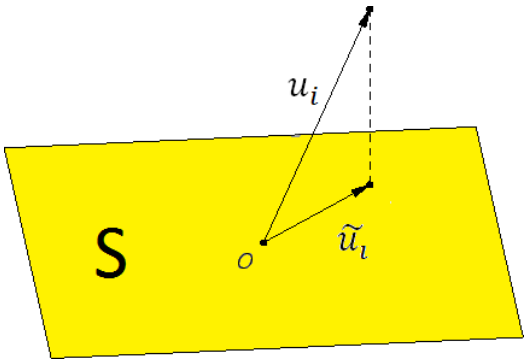

Figure 3. A typical 3D ortographic projection to a plane. The finite element method can be viewed as an ortographic projection in many dimensions.

The shape functions N_{Ai}, in general, do not depend on the index i. That is $N_{Ai} = N_A \ \forall i$. Therefore, Equation (1.3.10) can be written as:

$$\tilde{u}_i(x,y,z) = \sum_{A=1}^{N} c_{Ai} N_A(x,y,z) + g_i(x,y,z) \quad i = 1,2,3$$

Nevertheless, the inner product must depend on the index i. For a shape function N_A, we define:

$$B_A = \begin{pmatrix} N_{A,1} & 0 & 0 \\ 0 & N_{A,2} & 0 \\ 0 & 0 & N_{A,3} \\ 0 & N_{A,3} & N_{A,2} \\ N_{A,3} & 0 & N_{A,1} \\ N_{A,2} & N_{A,1} & 0 \end{pmatrix}$$

And the matrix:

$$K_{AB} = \int B_A^t K_E B_B \tag{1.3.11}$$

Where the matrix K_E is the same matrix we use in Equations (1.3.3) and (1.3.4). The integral is calculated for every entry of the 3×3 matrix K_{AB} [24].

The inner product $a(N_{Ai}, N_{Bj})$ is defined as the entry (i, j) of the matrix K_{AB}. Even for general functions W, V, it can be shown that this inner product definition is equivalent to [24]:

$$a(W, V) = W_{(i,j)} c_{ijkl} V_{(k,l)} \tag{1.3.12}$$

Where:

$$W_{(i,j)} = \frac{W_{i,j} + W_{j,i}}{2}$$

And the scalars c_{ijkl} are Hooke's law coefficients (Equation (1.3.2)) [37]. We now consider the difference $u_i - \tilde{u}_i$ and apply the orthographic property to obtain the scalars c_{Ai}. That is:

$$a(N_{Aj}, u_i - \tilde{u}_i) = a(N_{Aj}, u_i) - a(N_{Aj}, \tilde{u}_i) = 0$$

Using the Hooke's law for u_i (Equations (1.3.2), (1.3.3),(1.3.4) and (1.3.4)) we have that:

$$a(N_{Aj}, u_i) = a(N_A, u_i) = \int N_{A(i,j)} c_{ijkl} \epsilon_{kl} = \int N_{A(i,j)} \sigma_{ij}$$

Integrating by parts:

$$\int N_{A(i,j)} \sigma_{ij} = -\int N_{Ai} \sigma_{ij,j} + \oint_T N_{Ai} \sigma_{ij} n_j dT$$

But we supposed that u is the solution of the elastostatic equation. Therefore $\sigma_{ij,j} = 0$ and $\sigma_{ij} n_j = 0$ in T_H. These conditions simplify the equation system. The border conditions in T_G are different, because in this zone, the displacement is already known.

We suppose that $N_{Ai} = 0$ on T_G. This implies $a(N_{Ai}, u_i) = 0$. This forces g_i to satisfy the boundary conditions on T_G:

$$\tilde{u}_i(x, y, z) = 0 + g_i(x, y, z) \qquad \forall (x, y, z) \in T_G$$

Also, $g_i(x, y, z) = 0 \quad \forall (x, y, z) \notin T_G$. The inner product becomes:

$$a\left(N_{Aj}, u_i - \tilde{u}_i\right) = -a\left(N_{Aj}, \tilde{u}_i^m\right) - a\left(N_{Aj}, g_i\right) = 0$$

Where:

$$\tilde{u}_i^m(x, y, z) = \sum_{A=1}^{N} c_{Ai} N_A(x, y, z)$$

We get the equation:

$$a\left(N_{Aj}, \tilde{u}_i^m\right) = -a\left(N_{Aj}, g_i\right) \tag{1.3.13}$$

The Equation (1.3.13) applies for each N_{Aj}, obtaining a system of $3N$ equations. It is called the weak form of the elastostatic equation. When there are no restrictions on the space of shape functions the solution of the weak form is the same as the solution of the elastostatic equation [29,24]. The function g_i is a combination of shape functions that are not zero in T_G. We denote this space of functions by S_{T_G}.

$$g_i = \sum_{k \in S_{T_G}} q_k N_{ki}(x, y, z)$$

If we take the function $\tilde{u}_i^m$, expand it in Equation (1.3.13) and substitute the expansion of g_i we get:

$$a(N_{Aj}, \tilde{u}_i^m) = \sum_{B=1}^{N} c_{Bi} a(N_{Aj}, N_{Bi}) = -\sum_{k \in S_{T_G}} q_k a(N_{Aj}, N_{ki}) = -a(N_{Aj}, g_i) \tag{1.3.14}$$

This is the expanded weak form of the elastostatic equation. Despite these equations allow to obtain the scalars c_{Bi}, this set of equations is still semi-discrete because the inner products are made of continuous integrals. The complete discretization of the problem requires a precise definition of the shape functions and a method to calculate the inner products. This is described in the following section.

1.3.4 MESHING

The only restriction we imposed to the space of shape functions for the finite element method was $N_{Ai} = 0$ on T_G. In our three dimension model, the shape functions are defined as translations and dilations of the tensorial product of the hat function, which, in one dimension, is defined as:

$$H(x) = \begin{cases} 1 - |x| & \forall x \in [-1,1] \\ 0 & \forall x \notin [-1,1] \end{cases}$$

The hat function in three dimensions is defined as $H_3(x, y, z) = H(x)H(y)H(z)$ and in two dimensions as $H_2(x, y) = H(x)H(y)$ (Figure 4).

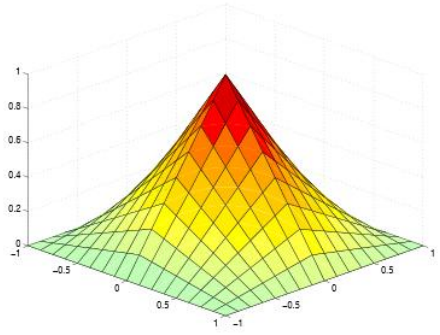

Figure 4. Two dimension hat function. The three dimension volumetric hash function cannot be plotted, but has analogous characteristics

We define the shape functions as:

$$N_A(x,y,z) = H_3\left(\frac{x-x_A}{\Delta x}, \frac{y-y_A}{\Delta y}, \frac{z-z_A}{\Delta z}\right) \tag{1.3.15}$$

The point (x_A, y_A, z_A) is called nodal point or node. The hat basis functions have the following interpolating property in each internal node:

$$\tilde{u}_i(x_A, y_A, z_A) = \sum_{K=1}^{N} c_{Ki} N_K(x_K, y_K, z_K) + g_i(x_A, y_A, z_A) = c_{Ai} N_A(x_A, y_A, z_A) = c_{Ai}$$

In the finite element method, the domain is subdivided into a grid of nodes, each one defining a shape function. This grid is arranged in sub domains, called elements which contain node arrays. In our three dimension model, these sub domains are built with 8 nodes, forming a tetrahedral domain.

The process of defining the elements and the nodes is called meshing [24]. In our case, our elements will be hexahedrons with eight numbered nodes (Figure 5).

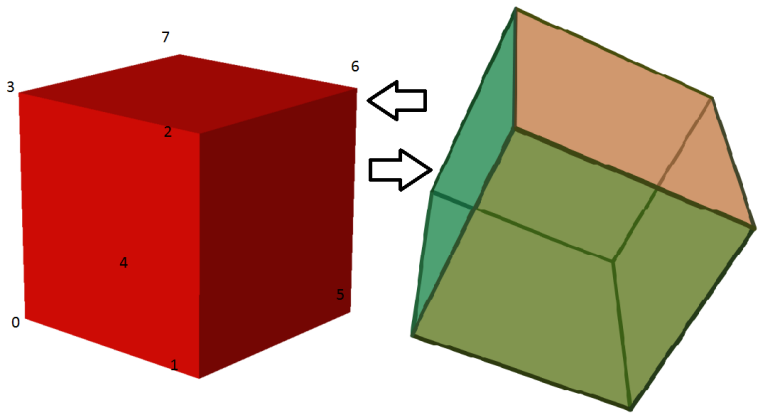

Figure 5. Hexahedral element equivalence. The jacobian makes a variable change to calculate the integral in a normalized element.

Consider a canonical element defined in the interval $[-1,1] \times [-1,1] \times [-1,1]$ with locally numbered nodes $\left(c_x^A, c_y^A, c_z^A\right)$ $A = 0,1,..,7$.

The node positions are numbered as in the red tetrahedron in Figure 5, centered in (0,0,0) and $\left(c_x^0, c_y^0, c_z^0\right) = (-1,-1,-1)$. In this canonical element, eight shape functions are not zero, and they are given by:

$$N_A(\varepsilon,\eta,\zeta) = \frac{1}{8}(1+\varepsilon_A\varepsilon)(1+\eta_A\eta)(1+\zeta_A\zeta) \tag{1.3.16}$$

Where ε_A, η_A and ζ_A can take the values ± 1 in all combinations. For example, $\varepsilon_0 = \eta_0 = \zeta_0 = -1$. In general, the elements will not have canonical form, but they can be transformed (Figure 5) facilitating the calculation of all inner products integrals, restricting them to the canonical element. In finite element method the inner product is obtained by integrating all inner products in each element sub-domain, and then, adding all results together.

The transformation to get the coordinates of a general element with nodes (x_A, y_A, z_A) $A = 0,1,\ldots,7$ is given by [24]:

$$x(\varepsilon,\eta,\zeta) = \sum_{A=0}^{7} N_A(\varepsilon,\eta,\zeta)x_A \qquad y(\varepsilon,\eta,\zeta) = \sum_{A=0}^{7} N_A(\varepsilon,\eta,\zeta)y_A$$

$$z(\varepsilon,\eta,\zeta) = \sum_{A=0}^{7} N_A(\varepsilon,\eta,\zeta)z_A$$

With the Jacobian matrix:

$$J = \begin{pmatrix} x_{,\varepsilon} & x_{,\eta} & x_{,\zeta} \\ y_{,\varepsilon} & y_{,\eta} & y_{,\zeta} \\ z_{,\varepsilon} & z_{,\eta} & z_{,\zeta} \end{pmatrix}$$

With these tools we can calculate the inner product between two shape functions N_A and N_B for all degrees of freedom on a particular element E.

After constructing the matrix of functions $B_A^t K B_B$ shown in Equation (1.3.11), then each entry f of this matrix is integrated on the canonical element, with the following change of variables:

$$\int_E f(x,y,z)dV = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} f\big(x(\varepsilon,\eta,\zeta), y(\varepsilon,\eta,\zeta), z(\varepsilon,\eta,\zeta)\big)|J^{-1}|d\varepsilon d\eta d\zeta \tag{1.3.17}$$

With the shape functions defined in Equation (1.3.15) and the change of variables of Equation (1.3.16) the result of Equation (1.3.17) can be calculated analytically. This finishes the process of the discretization of the elastostatic equation.

The integrals associated to the inner products can be obtained with the Gaussian Quadrature. This can be written as [29,24]:

$$\int_V f(x,y,z)dV = \sum_{s=1}^{P} w_s f(x_s, y_s, z_s) + E_P$$

Where E_P is the approximation error. This approximation is exact when the function $f(x,y,z)$ is a polynomial of a degree $n(p)$ that depends on the rule type. The eight point Gaussian Quadrature in three dimensions makes this relation exact ($E_P = 0$) for the shape functions described in Equation (1.3.16).

1.3.5 ONE ELEMENT PER VOXEL

The final step of the orthographic process described in Equations (1.3.13) is to produce a linear system of equations:

$$K\vec{c} = \vec{f} \tag{1.3.18}$$

Where K is a matrix of inner products that is also called Stiffness matrix. The vector $\vec{c}$ is the vector of scalars c_{Ai} and the vector $\vec{f}$ is called the force vector. The Equation (1.3.18) is called the model equation [24]. The size of this system of equations depends on the number of

shape functions (associated to the number of nodes and elements in the mesh model). The matrix K is very sparse. The number of elements different of zero per each line of the matrix is called the bandwidth of the matrix. The limited matrix bandwidth is a very useful property for an efficient computational representation of K [40,6].

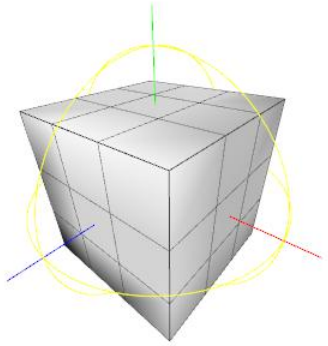

Figure 6. A 3×3 array of tetrahedral elements. The central element is surrounded by 26 elements.

As we show in Figure 6, according to the mesh model we described before, and the shape functions we are using, each element has at most 8 neighbors and each node is affected by only 27 neighbors (including itself). Therefore, the bandwidth of the matrix is $3 \cdot 27 = 81$. This can be the simplest approach to construct the mesh, taking advantage of the discretization of the proper image voxels. Every image voxel is considered an element, and each one of these elements is made of eight nodes. We can say at least three advantages of this meshing procedure:

1. The mesh is very easy to construct. There is no need to construct the mesh with computationally expensive procedures.
2. The mesh precision is directly associated with the discretization process in which the image was acquired
3. Each pixel color can be easily characterized by different Lame constants (different mechanical properties associated to each color).

The main disadvantage of this kind of mesh is that the number of elements and the size of the equation scales very quickly. For example, for a very small 3D image $50 \times 50 \times 50$ we have a system of $50^3 \cdot 3 = 375000$ equations. The stiffness matrix, with its bandwidth 81

will contain $50^3 \cdot 3 \cdot 81 = 30375000$ different real numbers (which requires a large amount of memory of $115MB$ just for the matrix). With a slightly larger image $100 \times 100 \times 100$ it is necessary to solve a system of 3 millions of equations, with a stiffness matrix with 243 millions of real numbers that requires a memory of 0.9 GB. In the next chapter we will describe how to overcome these computational problems.

1.4 UNIAXIAL COMPRESSION TEST

The final result of our FEM implementation is not the displacement vector for each node. The final result is a Lame constant, the Young module, obtained from the uniaxial compression test. To get this number, it is necessary to integrate the stress tensor on one of the faces of the rock sample.

This is done by passing through each element on the upper face, integrating the applied force using the displacement function at Equation (1.3.10) and the Hooke law of Equation (1.3.8). At the end, all forces are added together to get the total force necessary to deform the rock sample. The uniaxial stress is given by this force divided by the total upper part area of the rock. By using Hooke´s law shown in Equation (1.4.1) we can get the Young Modulus of the rock sample E. The equation is:

$$\sigma = E\epsilon \tag{1.4.1}$$

Where σ is called the uniaxial stress, and ϵ is the rock sample deformation. The constant E is called the Young modulus and it is a measure of rock resistance to deformation during linear behavior. When the value of ϵ is larger, the Young modulus is not a constant and the stress-strain relation is not linear. We show this tendency in Figure 7.

So, the rock sample is loaded along one of its axis, assuming linear behavior, producing a specified deformation [17]. In general the axis in which the force is applied is the Y-axis.

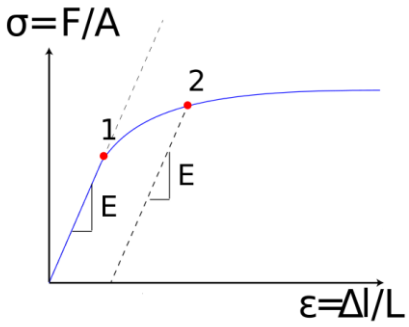

Figure 7. A typical stress-strain curve. The relation is linear, until the strain reaches the value 1. After value 2, the stress remains constant.

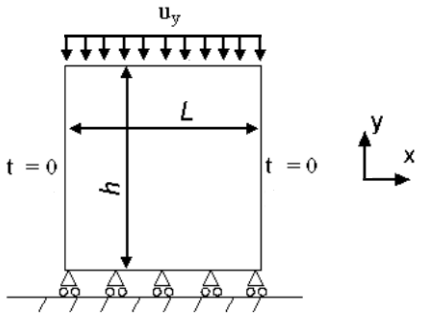

Figure 8. Uniaxial compression test. The boundary conditions impose zero displacement in the y-axis of the lower nodes, and a fixed displacement in this same axis in the upper nodes.

A schematic of the test is shown in Figure 8: A small displacement u_y is applied along the y-axis, applying a normal stress to the surface. There are no restrictions on material displacement, except in the lower part of the sample. Therefore, there will be a displacement of the material along z and x-axis. The total force applied to produce the controlled displacement along y-axis can be obtained and all data to get the Young modulus are known. We have:

$$\epsilon = \frac{u_y}{h}$$

Where u_y is the applied displacement and h is the rock sample height. We also have the total force applied F and if A denotes the upper area of the rock is sample, then:

$$E = \frac{\rho}{\epsilon} = \frac{F \cdot h}{A \cdot u_y}$$

CHAPTER 2. COMPUTATIONAL IMPLEMENTATION

2.1 WORKFLOW

In the following sections we describe our computational implementation of the numerical simulation of uniaxial compression test based on the finite element method. Most computational decisions are supported by adequate references, but other aspects are purely technical details and they are not justified (because it is not necessary). Some of these aspects are the file extensions (.rw3, .rw3,.bmm) and the module names. These are just a mechanism to classify our methodology and are used to identify our data workflow. They do not pretend to define a standard.

Almost all aspects of our workflow were implemented from scratch, using the C++ computer language and some basic CPU parallelization techniques. We do not pretend this to be the best implementation, but a new one in the set of existing software.

2.1.1 DATA PROCESSING

In this section we describe our computational workflow. It starts with a sequence of images and ends with a number. This number is the elastic module, whose value is justified by the numerical simulation. Our workflow is shown in Figure 9. Each step is associated with a file extension that facilitates a standardized format. Also, each step is associated to a small program or module. Our modules are:

- PIX: to handle basic digital image processing algorithms
- MESH: to construct a finite element method mesh
- FEM: to obtain the displacement vector for each node

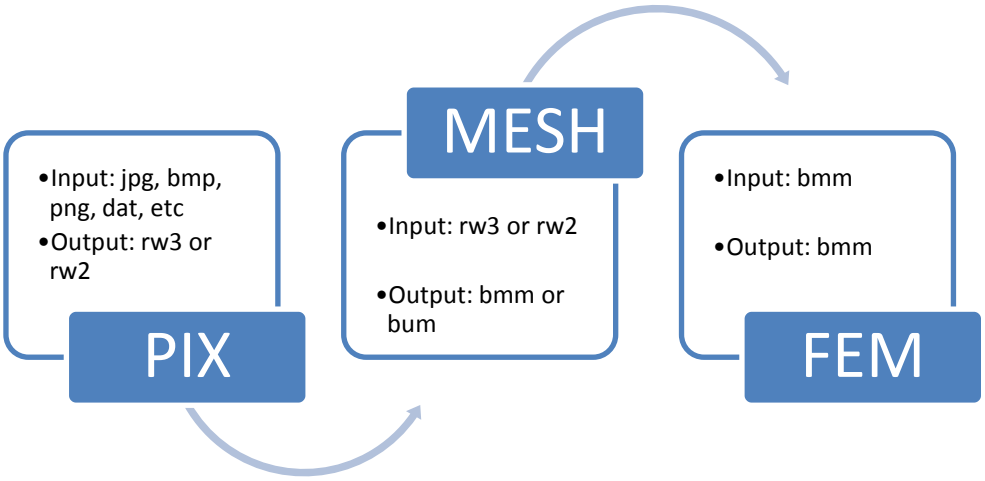

Figure 9. FEM processing modules and file extensions. Each one of these modules work as an independent computer program.

We use a cross-platform external library, SFML, to manipulate standard image formats: png, jpg,etc. OpenMP extensions are also used, to facilitate parallel processing. The chosen language is standard C++ and the code is compiled with GCC and Miscrosoft compiler.

2.1.2 MODULE PIX

As shown in Figure 9, the first step in this workflow consists in taking a sequence of images to construct a mesh. The input images can be interpreted as sequences of 1's and 0's, or as a sequence grayscale rectangular pixel arrays in which each color interval is associated to a different rock material (with its corresponding Lamé constants). After the grayscale array has been established, a mesh structure, suitable for finite element method, is constructed.

To start from a common point, we convert any kind of image sequence to a raw binary file, containing gray scale information, size and sample dimension. We use the extensions 'rw3' and 'rw2' for these file structures.

Table 1. File structure for extensions rw3 and rw2

TYPE	UNIT BYTE SIZE	LENGTH	DATA TYPE
Header			
Unsigned int	4	1	Dimension
Unsigned int	4	1	Width (W)
Unsigned int	4	1	Height (H)
Unsigned int	4	1	Depth (D)
Data			
Unsigned char	1	Width × Height × Depth	Image color

We show in Table 1 the structure of these raw files. The main data is the image's color information, which is simply a sequence of unsigned char types. It represents a gray tone in the integer interval $\{0,1,2,\dots,255\}$. Each row of voxels is stored contiguously, and later, each image is stored contiguously. Therefore, the color at the coordinate (x, y, z) of the image is located in the file position $c(x, y, z)$ of the binary file list, where:

$$c(x, y, z) = x + y \cdot W + z \cdot W \cdot H$$

Where W is the image width and H the image height. The module PIX contains simple image processing functions and is capable of calculating rock porosity by counting the number b of voxels whose color is less or equal to an upper bound L. PIX calculates porosity P_r as:

$$P_r = \frac{b}{W \cdot H \cdot D}$$

Where the values W, H and D define the image resolution (Table 1). PIX can also take sub volume samples from the original image, reduce the image size using linear average and create study cases based on spherical pores. To produce a smaller version of a 3D image whose original size is (W, H, D) PIX only calculates averages. If the resized image size is (w, h, d) where $W = n_x w$, $H = n_y h$ and $D = n_z d$, then, the color of the voxel (i, j, k) of the smaller image is the average of the voxels located in the box limited by the vertices $\left(n_x i - \frac{n_x}{2}, n_y j - \frac{n_y}{2}, n_z k - \frac{n_z}{2}\right)$ and $\left(n_x i + \frac{n_x}{2}, n_y j + \frac{n_y}{2}, n_z k + \frac{n_z}{2}\right)$ in the larger image.

Table 2. File description for extensions bmm

TYPE	UNIT BYTE SIZE	LENGTH	DATA TYPE
Header			
Unsigned int	4	1	Space dimension S ($\mathbb{R}^2$ or $\mathbb{R}^3$)
Unsigned int	4	1	Number of degrees of freedom D
Bool	1	1	A flag that tells if the model is solved
Unsigned int	4	1	Number of nodes per element T
Unsigned int	4	1	Total number of nodes N_N
Unsigned int	4	1	Total number of elements N_E
Unsigned int	4	3	Image resolution ($Width \times Height \times Depth$)
Double	8	3	Sample real dimensions
Node list. Repeat for every node, N_N times			
Int	4	1	Node index
Double	8	S	Node position coordinates
Unsigned int	4	1	Node flags (Boundary conditions)
Double	8	S	Boundary conditions
Double	8	S	Node field conditions
Double	8	D	Node displacement
Element list. Repeat for every element, N_E times			
Int	4	1	Element index
Int	4	T	Node indices contained in the element
Int	4	1	Coefficient index of the matrix associated to the element
Unsigned int	4	1	Element flags (position in the array)
Stiffness matrix description (only once)			
Unsigned int	4	1	Number of rows for each matrix R
Unsigned int	4	1	Number of columns for each matrix C
Unsigned int	4	1	Total number of stiffness matrices N_M
Stiffness matrix list. Repeat for every matrix			
Unsigned int	4	1	Matrix index
Double	8	$R \times C$	Matrix entries

2.1.3 MODULE MESH

Once a file with extensions “rw2” or “rw3” is produced, an input mesh for the finite element method should be generated. Once again, we describe the mesh with raw binary files, using extensions “bmm” and “bum” to distinguish them. We associate the extension “bmm” to any kind of mesh or simplicial complex, usually made of tetrahedrons [2]. The extension “bum” is associated to a file which takes advantage of the mesh uniformity (if every voxel is an element, then all elements are of the same shape and size). The program “MESH” takes a 3D or 2D file as described in Table 1 and produces a model, as described in Table 2. Even though there are other standardized formats to store Finite Element Meshes in files, we use this simple and straightforward approach adapted to our needs [36].

We note that the structure shown in Table 2 can be used to describe any kind of mesh, regular and non-regular. When the mesh consists of a regular structure of elements, the node position can be obtained simply from the position of its parent elements. Therefore, the position of a node can be characterized by an integer index. This concept is applied in our file extension “bum”, in which the sequence of real numbers that characterize the node position is replaced by an integer index.

2.1.4 MODULE FEM

The program FEM takes the mesh file with extension “bmm” (or “bum”) and produces a file with the same structure, but with the displacement field solved. The information contained in an unsolved “bmm” file is enough to apply the numerical finite element method to solve the node displacement and this information can be stored in the same file. In other words, FEM takes an unsolved “bmm” file as input and produces a solved “bmm” file. This is the most resource and time consuming module in the workflow.

When FEM reads an input “bmm” file, it passes through each one of the elements, calculating the inner product of the associated shape functions. These inner products are assembled in the large sparse matrix K, shown in Equation (1.3.18). Also, the node displacements are enumerated and assembled in a single vector of unknowns $\vec{c}$. Once the

system $K\vec{c} = \vec{f}$ is mounted, the linear algebra numerical method of the conjugate gradient is used to solve the linear system [40]. The method is convergent and applicable because the matrix K is symmetrical and diagonally dominant [40,24].

Specifically, the method we chose to solve the linear system is the conjugate residual with Jacobi preconditioner [40]. The solved file is constructed with the enumerated nodes taking its displacement value from the vector $\vec{c}$.

With the node displacement, a new mesh model can be produced where the position of the each node k is given by:

$$u_i^k = u_i^k + \Delta\, u_i^k$$

Where $\Delta\, u_i^k$ is the calculated node displacement. For each degree of freedom i, the value of $\Delta\, u_i^k$ is contained in the displacement vector $\vec{c}$ or in the boundary conditions T_G. (Equation (1.3.6)). These new positions provide a new model of the rock after the compression test (Figure 10). We remark that elementary cubic voxels can be deformed after the displacement, and they do not necessarily maintain its cubic shape.

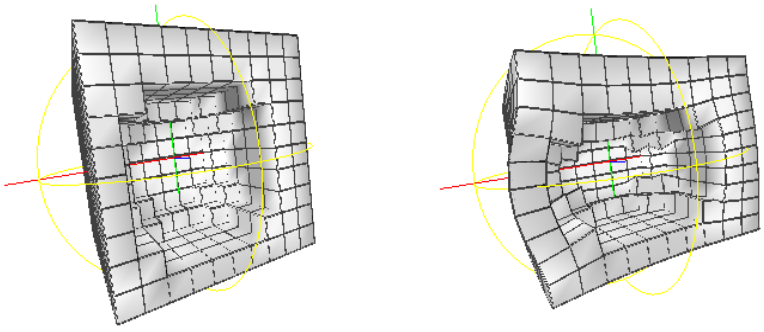

Figure 10. A simple rock sample mesh before and after deformation. The second mesh is obtained by adding the displacements $\Delta\, u_i^k$ to each node position. We note that the hexahedral elements can lose its initial symmetrical shape.

The symmetric "bum" files are processed by another module which has the same functions of FEM but is optimized for symmetry (using the element by element technique,

which we describe later). We call this module BUMFEM and it is equivalent to FEM in all its functionality.

2.1.5 MODULE DP

Calculating petrophysical constants is only a matter to take a solved 'bmm" file and interpret its results. This is done in an external program called "DP" (which does not appear in Figure 9). This program makes a numerical integration in the upper part of the rock sample, calculating the necessary force to establish the displacement. The Young module can be obtained by dividing this force between the area of the sample and the deformation rate.

2.2 MEMORY AND NUMERICAL ISSUES

2.2.1 MATRIX REPRESENTATION

In our C++ implementation, we use several namespaces to limit the scope of the different matrix types. These namespaces are "sparse", "full" and "sm" (small). They encompass the three kinds of matrices (and vectors) we use in our code:

1. Small matrices and vectors, whose function is to calculate local inner products per element, establish node positions and store Jacobian matrices.
2. Large sparse matrices to iteratively solve the large main system of the model: $K\vec{c} = \vec{f}$.
3. Large full vectors whose function is to store nodal displacements (vectors $\vec{c}$ and $\vec{f}$ of the system $K\vec{c} = \vec{f}$).

We do not use large full matrices in our code.

The small matrix and small vector information is stored in a simple adjacent memory array of size N. The value of N is the memory required to store them. We suppose this number is never very large. For example the elementary Lamé constants matrix size is 6×6 ($N = 36$). As we suppose the memory these kinds of matrices and vectors use is small, then it does not require to be shared. Each small matrix and each small vector frees and assigns its

own memory, and the copy operator simply copies explicitly one memory zone to another. These matrices must overload common operators for matrices like addition, difference, scalar product, matrix product, matrix-vector product, determinant, transposition and inversion.

On the other hand, large full vectors store a lot of information. Their size N is very large and every entry must be explicitly stored. In our algorithms, copies and redefinitions are common for these kinds of vectors [15]. For example let's consider the following redefinition for vectors $\vec{r}, \vec{a}, \overrightarrow{r_p}$ and a scalar c:

$$\begin{aligned} \overrightarrow{r_p} &= \vec{r} \\ \vec{r} &= \vec{r} + c\vec{a} \end{aligned} \tag{2.2.1}$$

We use these redefinitions in the conjugate gradient algorithm [40]. A naive implementation of this would do the following: first, the vector $\vec{r}$ is assigned to vector $\overrightarrow{r_p}$ executing a copy operator. Then, the result of $\vec{r} + c\vec{a}$ (which involves two copies) is reassigned to vector $\vec{r}$. A total of 4 copies are performed in these simple two steps. If we handled these copies the same way as small vectors and matrices, then these simple operations would require to pass large amounts of data from one memory zone to another, affecting performance. To avoid this, we overload the copy operators for full vectors. When a vector is assigned to another then, only its array pointer is copied. This adds the possibility that the same array pointer coexist within several vectors and therefore it is necessary to have a control about the number of copies in which each pointer is shared. This is done using an external host of pointer references.

When a vector is modified and its data pointer is shared, a copy is unavoidable. In this case, a new array is created inside the local vector, and all current information copied. Only after this, the data modification is executed. To have a strict control over memory copies and modifications, full vectors have access operators that handle these situations.

Using these copy operators and defining functions for arithmetic common operations, the number of copies in Equation (2.2.1) is reduced to only one. The operators in Equation (2.2.1) work in the following way:

1. Copy operator: $\overrightarrow{r_p} = \vec{r}$. The data pointer contained in $\vec{r}$ is copied to $\overrightarrow{r_p}$. Now the two vectors share the same unique data.
2. Arithmetic operator: $\vec{r} = \vec{r} + c\vec{a}$. This is done through a call to a unique function that handles operations of type $c_1\vec{r} + c_2\vec{a}$. This is an access operator which requires scalars c_1 and c_2 and a reference to vector $\vec{a}$. Before modifying its own data, vector $\vec{r}$ "notes" that its data is being shared with another vector, and therefore proceeds to copy its data to a new local pointer. After doing this, the operation is executed on the new data pointer.

It is not necessary to check shared data if the vector data is being read. In case of Equation (2.2.1), vector $\vec{a}$ is not affected by this overhead. This is a consequence of the shared memory architecture [8].

In our finite element method, only a big sparse matrix object is required. This is the limited bandwidth matrix K of the system of Equations (1.3.18). The most important operator for this kind of matrix is the product of matrix by vector. It is necessary for the iterations of the conjugate gradient method that ultimately solves the system $K\vec{c} = \vec{f}$. To optimize this product, the data in this matrix is arranged in a real number memory block of length $B \cdot M$ where B is the matrix bandwidth and M is the matrix number of rows.

The column position of these matrix entries is stored in an integer memory block of the same size. Each index row is sorted, in order to facilitate its search (in logarithmic complexity) during insertion of new terms. All matrix access operations are based on the Binary Search Algorithm [10]. The memory can be divided in M rows, each one containing B slots to enter an index and a real number. We show these processes in Table 3.

For a bandwidth B, the complexity of reading operators per row is $O(log\, B)$ and the insertion is at most $O(B)$. This is highly efficient because, in our case, the number B is 81. A typical search will require at most 8 iterations. The efficiency of these operations is important to write data on the matrix, when it is being assembled with the inner product of functions in the finite element method.

Table 3. Sparse matrix access operators

<table>
<tr><th>Operation</th><th colspan="3">Method</th></tr>
<tr><td>Creation</td><td colspan="3">The matrix is created assigning 0 to all real entries and -1 to all index entries. This negative value indicates that the slot is empty</td></tr>
<tr><td rowspan="3">Insert a non zero value in position (i,j)</td><td rowspan="3">Use binary search to find index j at row i</td><td colspan="2">Index found. Replace current value with new value</td></tr>
<tr><td rowspan="2">Index not found. Use binary search to find first empty slot, an index with the value -1, but with a preceding index positive.</td><td>Slot found: According to the value j, shift current indices to preserve order. After this, insert value</td></tr>
<tr><td>Slot not found. Bandwidth full. Create a new copy of the matrix with larger bandwidth</td></tr>
<tr><td rowspan="2">Insert zero in position (i,j)</td><td rowspan="2">Use binary search to find index j at row i</td><td colspan="2">Index found. Translate back all next indices in the row, filling with -1 the last positive index</td></tr>
<tr><td colspan="2">Index not found. Do nothing</td></tr>
<tr><td rowspan="2">Return value in position (i,j)</td><td rowspan="2">Use binary search to find index j at row i</td><td colspan="2">Index found. Return associated value</td></tr>
<tr><td colspan="2">Index not found. Return 0.</td></tr>
</table>

2.2.2 PARALLELIZATION

Almost all FEM operations are involved with matrix operations. Therefore, it is important, not only to optimize its memory representation but its arithmetic operators as well. The multi-core architecture, nowadays common in many computers, allows simple parallel application development and OpenMP provides its language extensions and tools. There are other technologies that provide more scalability (like MPI) but OpenMP provides more control and transparency [8,38].

As our simulations will be run, in most cases, in common workstations, we choose OpenMP. Another level of parallelism must be considered, exploiting Streaming SIMD Extensions, a technology supported in most modern compilers and microprocessors [8].

Therefore, we identify two levels of CPU parallelization:

1. SIMD parallelization: Single instruction multiple data. Based on SSE2 instructions.
2. MP parallelization: based on OpenMP, it exploits multi-core processors

There are several steps in the Finite Element Method implementation that can exploit both kinds of parallelization, others steps can handle only one and others cannot use any of them.

One of the first limitations of the SIMD parallelization is that it can only be applied with maximum efficiency in aligned memory blocks. This means that the associated pointer in which an arithmetic operator is applied must be a muiltiple of $16 = 0x000010_{HEX}$. This can be easily done by defining a template ´scalar´ which is a single or double precision floating point number with the compiler specific directive that aligns its declaration.

The SIMD SSE2 operators effectively calculate a sequence of 2 or 4 floating point operations in only one function call. All small matrix operations (described in Section 2.2.1) can make use of these extensions. This reduces the number of flops in every matrix operation by a factor of 4 (single precision) or 2 (double precision). MP parallelization is not applicable on small matrices because the overhead of creating and using the associated threads does not compensate parallelization in small dimensions. SIMD parallelization optimizes:

1. Small matrix and vector addition and subtraction;
2. Matrix by matrix products.

Matrix transposition, determinant, solution of linear systems (using Cramer´s rule) and matrix by vector product are not SIMD parallelized because they do not access adjacent memory.

Matrix operators in large vectors can be parallelized in both levels. Addition, subtraction and sparse matrix by vector operators are distributed across the machine processors and the basic internal operations are parallelized because its memory is adjacent. For example, if sized N vectors $\vec{a}$ and $\vec{b}$ are added and the result assigned to $\vec{r}$, the pseudocode is shown in Figure 11.

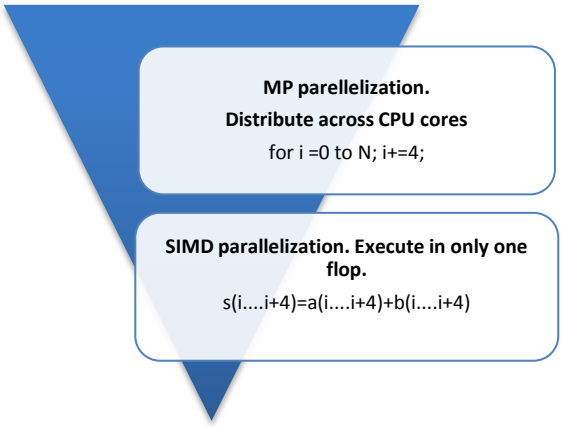

Figure 11. Parallelization of vector addition. This kind of parallelization is easy to define in MP and SIMD level.

MP parallelization is applied across T threads, then each thread only has to execute $\frac{N}{4T}$ flops in single precision, using SIMD parallelization.

Reduction operators such as dot product are parallelized by assigning an auxiliary variable to each thread. The thread executes only one section of the dot product additions. Each variable is updated independently and at the end, these auxiliary variables are reduced to only one. If T is the number of threads, we show the dot product operation $\vec{a} \cdot \vec{b}$ in Figure 12.

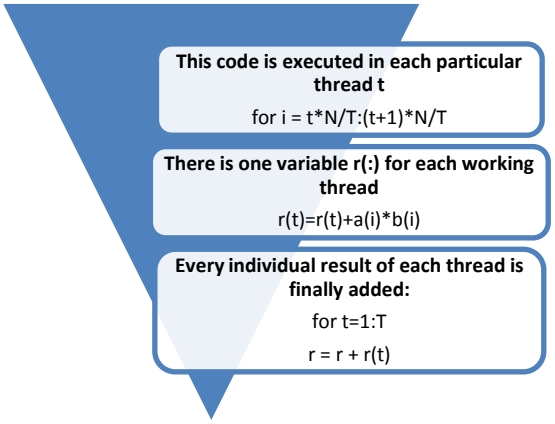

Figure 12. Parallelization of vector dot product. This is a common reduction operator which is parallelized with an auxiliary variable per thread.

All other basic vector operations can be described as variations of these two basic operations. Step 2 in Figure 12 is reduced even more by using SIMD parallelization. The products $a_i \cdot b_i$ are calculated in groups of four (for single precision numbers), but the additions must be executed one by one.

The heaviest parallel operation in the finite element method is matrix by vector product. This is necessary to execute the conjugate gradient method to solve the system $K\vec{c} = \vec{f}$. As we discussed earlier, the matrix K, whose size is $N \times N$, is stored as an array A of size $N \times B$ where B is the bandwidth of the sparse matrix. The function that associates each element of this array to its corresponding entry in K is denoted by l_f. This means that if j is the memory position of row i, then $l_f(i,j)$ denotes the real column in the row i of the matrix K in which the entry is located. This is because in general, memory position and real position do not match.

The MP parallelization distributes the job of multiplying scalars across the rows of matrix K, and the SIMD parallelization decreases, in performance, the size of the bandwidth B. This is shown in Figure 13. The group $v(l_f(i, j \ldots j+4))$ is not adjacent in memory, but this can be solved with an additional local copy.

The process of mounting the matrix K is associated with the calculation of inner product functions in Equations (1.3.13). These inner products are used to update and modify the entries of this matrix. The natural mechanism is to parallelize the process in groups of elements. Each element produces a modification to K, and this is a source of memory conflict.

To avoid this conflict, the main inner product process (numerical integration, matrix products, etc) are processed inside each thread, but the update is done inside a critical zone [8]. To reduce the performance impact of this, we take into account the optimizations shown in Table 3. This is a MP parallelization.

The complexity in memory access for numerical integration and inner products evaluation makes it difficult to optimize with SIMD parallelization and therefore it is not applied. Nevertheless, it is used internally by the small matrices and their basic operators.

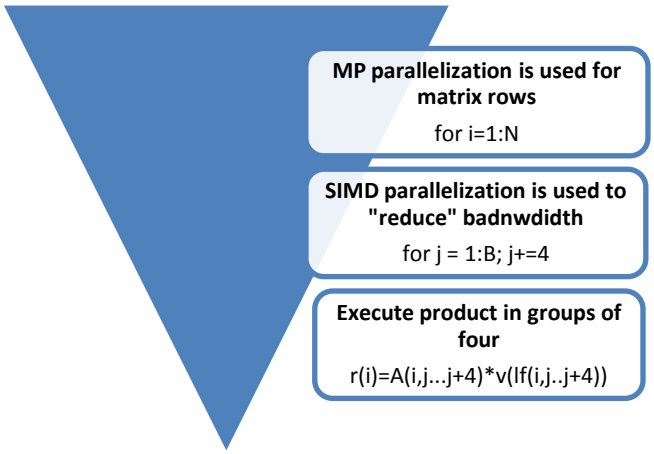

Figure 13. Parallel sparse matrix – vector product. The MP parallelization is applied in matrix rows and the SIMD parallelization is used to reduce the number of flops associated to the bandwidth.

2.2.3 SINGLE AND DOUBLE PRECISION

The code of all programs shown in Figure 9 can be compiled to use single precision floating point arithmetic or double precision floating point arithmetic. The first thing to take into account is the limited precision of both schemas which only produce rational numbers [15]. The basis on both representations is 2, which means that a number r is expanded as:

$$r = \left(c_1 \frac{1}{2} + c_2 \frac{1}{2^2} + \cdots + c_m \frac{1}{2^m}\right) 2^E \tag{2.2.2}$$

Usually $m = 23$ and $E = 8$ for single precision representation. Another bit is used to distinguish positive and negative numbers, giving 32 bits (4 bytes). If a difference between two numbers is smaller than $\epsilon \approx 1.2 \cdot 10^{-7} \approx \left(\frac{1}{2}\right)^{23}$ then it will not be adequately represented by Equation (2.2.2) [26]. Taking this into account, we consider that two real numbers in single precision are equal if their absolute value difference is smaller than $\epsilon = 1.2 \cdot 10^{-7}$. This means that single precision will provide roughly 6 correct digits.

For double precision, we have $m = 53, E = 11$. In this case $\epsilon \approx 1.1 \cdot 10^{-16}$, so we take $\epsilon = 1.2 \cdot 10^{-16}$. We get 15 correct digits.

The choice between double and single precision is not obvious. In our examples, we are considering very large vectors and finite element mesh models. Double precision representation requires 4 more bytes, which means an important memory issue, not for a simple 3D vector representation, but for thousands of entries in a large full vector. In Section 2.3 we illustrate the precision differences with an example. In general it is less than 1%, which is not representative in most practical cases (Chapter 3).

The code is parallelized in a better way using single precision number representation. The SIMD commands will execute four operations simultaneously, which will produce a slightly faster program. Despite the conjugate gradient method to solve the system shown in Equation (1.3.18) uses MP and SIMD parallelization, this is the slowest and most computationally intensive part of our program. This is due to the number of iterations that must be applied to get the solution vector and the large size of the matrices involved.

2.2.4 CONJUGATE GRADIENT METHOD

To solve the system of Equation (1.3.18) we use a variation of the conjugate gradient method, called the Conjugate Residual. According to [13] this method is preferred for physical data, when it is not justifiable to seek a solution beyond its accuracy. The solution should match the error of the input data and the design of the upper limit θ. The algorithm is shown in Table 4.

Table 4. Conjugate residual method

$x = 0;\quad r = c;\quad s = Kr;\quad \rho = r^t s;\quad p = r;\quad q = s;$

$while\quad \|r\| < \theta$

$\alpha = \rho / \|q\|^2$

$x \leftarrow x + \alpha p$

$r \leftarrow r - \alpha q$

$s = Kr$

$\bar{\rho} = \rho, \rho = r^t s$

$\beta = \rho / \bar{\rho}$

$p \leftarrow r + \beta p$

$q \leftarrow s + \beta q$

We try to approximate the absolute difference $\|x - x^*\|$ where x^* is the solution of the system. If we suppose all data of matrix K has a tolerance ϵ, then we can say that:

$$\|\epsilon(x - x^*)\| \leq \|K(x - x^*)\| = \|Kx - c\| = \|r\|$$

We guarantee that the difference $\|x - x^*\|$ in displacement is inside the tolerance ϵ if $\|r\| < \epsilon^2\|x^*\| \approx \epsilon^2\|x\|$.

This is our convergence criterion, because this implies that:

$$\frac{\|x - x^*\|}{\|x\|} < \epsilon$$

The adequate choice of ϵ depends on the mechanical parameter to be determined. This is not always easy. We show in Figure 14 a typical curve of the normal force on a rock sample, changing with the number of iterations. This curve was produced with the sample S1, we use in the Chapter 3.

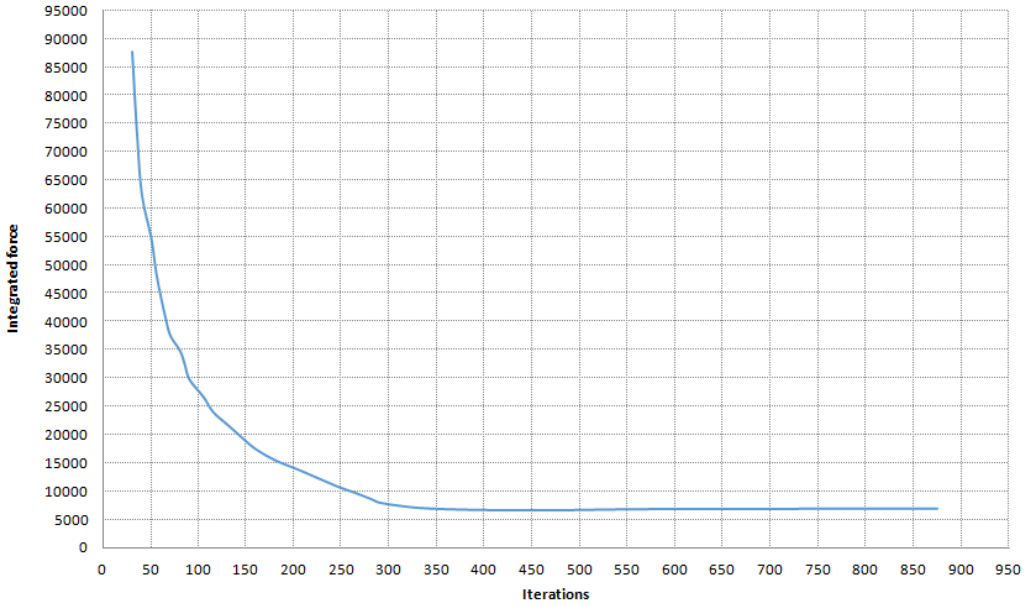

Figure 14. A typical normal force vs iterations. A very small percentage of the matrix rows is usually an enough number of iterations. The force is displayed in Newtons.

We should try to reach the curve when it becomes almost constant. As the petrophysical parameters can be diverse, we use an empirical rule in which the iteration process can be stopped after a certain number of iterations, and then check the variation of the estimated parameters. As shown in Figure 14, the stopping criterion should be surpassed, but never anticipated.

2.3 CODE VALIDATION

The easiest way to validate the code is to take a non-porous model of a rock. It is completely uniform and all elements have the same Lame constants. In this section we give an example about how to input data in the finite element method. We take the following constants:

$$E = 5.99 \cdot 10^{10}\ N/m^2 \qquad (2.3.1)$$
$$\upsilon = 0.40$$

An average of silica common ceramics [33]. These two constants are enough to define the other Lamé constants [37]. We first multiply E by a constant that nullifies its units. As E contains length and force, we define:

$$C_L = 10^{-5} m$$
$$C_F = N^{-1}$$

We define:

$$E_a = E C_L^2 C_F = 5.99$$

To define the matrix K in Equation (1.3.3) we must obtain λ and μ Lamé constants:

$$\lambda_a = \frac{E_a \upsilon}{(1+\upsilon)(1-2\upsilon)} = 8.557142857$$
$$\mu_a = \frac{E_a}{2(1+\upsilon)} = 2.139285714$$

We apply these Lamé constants to a regular element array of size $8 \times 8 \times 8$ (Figure 15). After all the workflow shown in Section 2.1 we recover the value:

$$E_a = 5.9899$$

To recover the original units we should multiply this result by $C_F^{-1} C_L^{-2}$. This is solved exactly after 63 iterations in the conjugate gradient method. The relative error of this procedure is less than 0.001%. The sources of this error are diverse: rounding procedure for the finite element definition, single precision representation and numerical integration.

If double precision is used to solve this same problem, the precision errors is almost imperceptible (less than 0.00034%) but the choice between single and double precision should not be based only on this criteria. Memory issues could be more important, especially for large examples (Section 2.2.3). Besides, in this example, reaching the solution with 15 correct decimals, took 97 iterations in the conjugate gradient method (an increment of 35%).

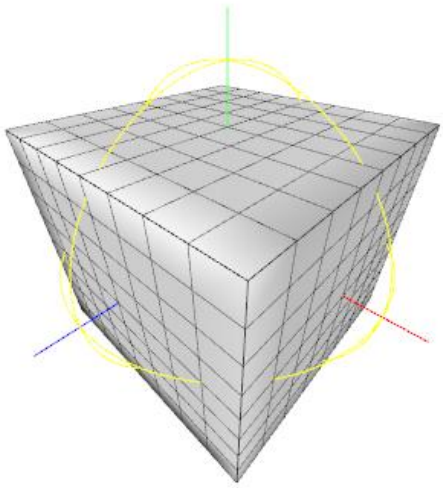

Figure 15. An $8 \times 8 \times 8$ array of uniform non porous elements.

2.4 THE STIFFNESS MATRIX PROBLEM: ELEMENT BY ELEMENT

One of the most important issues in our current finite element method is the stiffness matrix K. Despite its bandwidth is relatively small, as we discussed in Section 1.3.4, its size can be enough to easily consume all computer memory.

To avoid this, we can suppress the representation of the matrix, by making the matrix-vector product $K\vec{x}$ in the mesh model, element by element, without representing the stiffness matrix.

This can be a formidable task, from a computer point of view, but in our mesh representation we can take advantage of two key characteristics:

- All elements in the mesh are of the same size.
- The matrix of inner function products by element is the same, and only differs by the phase material

This means that the inner products per element must be calculated only once. The procedure of solving the system $K\vec{c} = \vec{f}$ in the Element by Element technique [24], consists in multiplying the vector $\vec{x}$ in each iteration in each one of the elements. Instead of storing the inner products in the global stiffness matrix, they are applied directly to the vector. This happens for every matrix product $K\vec{x}$. As we said, this would be a formidable computational task if the elements were not the same size and the inner product matrices were not calculated previously.

To parallelize this product, each thread takes control of a subset of elements. Each one of these elements changes the vector $\vec{x}$. This means that each thread has to update the same vector, and this can produce collisions. Using a critical zone for each vector update is a recipe for bad performance, considering the large number of elements that are being used (according to our tests, the critical zone costs about 20% to 130% more time per iteration, depending on the number of processors). We prefer to define one vector $\overrightarrow{x_k}$ for each thread.

At the end, the resultant vector $\vec{x}$ is given by:

$$\vec{x} = \sum_{k=1}^{T} \overrightarrow{x_k}$$

For vectors of very large size and a big number of possible threads, this can produce memory issues (because a new vector must be constructed for each thread). We prefer, then, to exploit less computer processors and maintain this architecture. This is not a waste, because in many modern computers, the number of processors is duplicated by hyper threading.

Our parallel element by element finite element method is significantly slower than the direct method, because of the large number of elements. On the other hand it requires significantly less amount of memory and it becomes the only alternative for very large rock models (more than $300 \times 300 \times 300$ elements).

CHAPTER 3. NUMERICAL EXPERIMENTS

3.1 MECHANICAL PROPERTIES AS FUNCTION OF POROSITY

3.1.1 EXACT RELATIONS

Our porous rock samples are modeled as two or more phases of material types bonded together along with empty pore space. Pores can also be modeled as another material type whose mechanical properties are zero. A naïve computational implementation can model their geometry in this way, but to avoid unnecessary calculations, we prefer not to include its associated voxels in the adjacency FEM mesh structure. The pore is modeled as real empty space. As we described in Section 1.2, each color in the microtomographic image sequence can be associated to a different phase. The number of gray tones the module PIX can handle is at most 255, which is fair enough in most cases where only two phases are considered [22]. We also can assume that each phase is isotropic and uniform, specifying these properties in the elementary matrix K_E at Equation (1.3.11). Yet, our FEM module does not restrict the choice of this matrix.

In general, real rock microtomographic images do not have restrictions on the shapes of the material or pore inclusions, which makes it impossible to get a general analytical solution for the internal stress field as a function of porosity; except for simple cases and geometries which could never be produced in a natural process of rock formation [22,19,32,25]. Despite this limitation, analytical boundaries or approximations to Young module as a function of porosity have been produced along many years of research, providing elastic module approximations with a relative difference of less than 20% respect to their real or FEM simulated counterparts, for a certain range of porosities and arbitrary pore shape [9].

The most precise upper and lower limit for the Young modulus in a material with several phases are the Voigt and Reuss limits [30]. If the rock sample contains n phases with Young modules $E_1, E_2, \dots, E_n$ and porosities $f_1, .., f_n$, then the maximum and minimum Young modules E_{min}, E_{max} satisfy the following equations:

$$E_{max} = f_1 E_1 + \cdots + f_n E_n$$

$$\frac{1}{E_{min}} = \frac{f_1}{E_1} + \cdots + \frac{f_n}{E_n}$$

The composed Young module can easily reach these upper and lower limits, using the configuration shown in Figure 16. They represent two kinds of material immersed inside the rock sample (with different mechanical constants). In both cases, the blue phase occupies $1/8$ of the total volume of the area of the image. The same pattern is repeated along Z-axis, providing the volume shape. The left configuration gives the Young modulus value E_{max} and the right one gives E_{min}. These configurations are not realistic and are never presented in natural formed rock samples.

Figure 16. Configuration of material types. These patterns are repeated along the Z-axis, producing a voxelized image in which 1/8 of its volume is blue and the rest is yellow material.

Let us suppose we have only two phases E_1, E_2, with $E_2 = 0$. This can be used as a simplified model for porous media. In this case $E_{min} = 0$ and $E_{max} = f_1 E_1$. This allows us to conclude the following:

1. The porous rock samples (with one of the phases equal to 0) produce a large imprecision in Voigt and Reuss limits.

2. The rock sample Young module is smaller than the phase Young module. We should expect that this tendency is accentuated according to porosity (given by $f_P = 1 - f_1$).

We validate our code using analytical estimates provided along many years of research, in which the Elastic Coefficients are expressed as functions of porosity [22].

3.1.2 DILUTE SPHERICAL INCLUSIONS

One of the few three dimension exact analytical solutions for simple pore geometries is given by Hashin [18,14]. The rock material is supposed to be biphasic, with Lamé constants λ_S, μ_S for the solid phase and λ_P, μ_P for a set of spherical inclusions.

With these constants, we can calculate the following Lamé constants [37]:

$$E_k = \frac{\mu_k(3\lambda_k + 2\mu_k)}{\lambda_k + \mu_k}$$
$$K_k = \lambda_k + \frac{2}{3}\mu_k$$

For $k = S, P$. The result of Hashin [18] establishes that the bulk and shear modulus of the sample are given by the following linear relations:

$$K = K_S + pP^{SP}(K_P - K_S) \qquad (3.1.1)$$
$$\mu = \mu_S + pQ^{SP}(\mu_P - \mu_S)$$

Where:

$$P^{SP} = \frac{3K_S + 4\mu_S}{3K_P + 4\mu_S}$$
$$Q^{SP} = \frac{\mu_S + F_S}{\mu_P + F_S}$$

And p is the porosity. Also:

$$F_S = \frac{\mu_S}{6} \cdot \frac{9K_S + 8\mu_S}{K_S + 2\mu_S}$$

The Young modulus of the sample is obtained using these constants:

$$E = \frac{9K\mu}{3K + \mu}$$

According to this equation, the Young module is not linear with porosity but the bulk and shear modules are. The physical interpretation Hashin assumes for this result are the following:

1. The inclusions are spherical
2. The inclusions are distributed uniformly inside the matrix (a perfectly isotropic pore distribution).
3. The strain tensor in any inclusion is not affected by the strain of the others. Their behavior is like if they were inside an infinite matrix body.

The last two points are equivalent to say that the pore inclusions are diluted inside the matrix. To model these strong conditions, we take a completely uniform distribution of pores inside a matrix with Lame constants $\lambda_S = 3, \mu_S = 0.5$. The pore constants are: $\lambda_P = \frac{\lambda_S}{3}, \mu_P = \frac{\mu_S}{2}$. These magnitudes are multiplied by $10^{10} N/m$, but this factor suppressed. This is a way to nullify dimensions in the Finite Element method. We give more details on this procedure in the next sections. We take an array of 27 pores and the distance between them is the same in all directions. This is a small, unrealistic perfect dilute rock model. The WorkFlow shown in Figure 9 is capable of providing these simple cases, by approximating the spherical pore by a voxelized image (Figure 17). The definition of pores, their position and the size of the image are specified in a text file whose structure is shown in Table 5. The program PIX generates an "rw3" three dimension image based on these specifications.

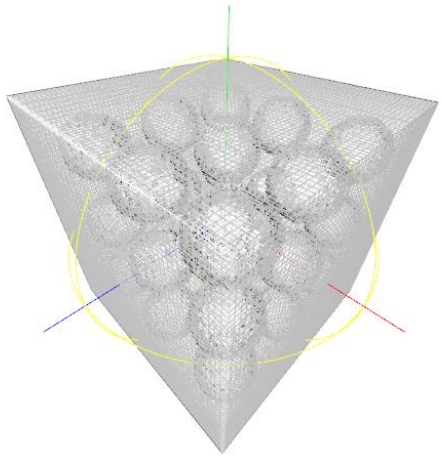

Figure 17. Spherical uniform array. The spherical inclusions model a perfectly isotropic dilute rock sample. The mesh resolution is $100 \times 100 \times 100$

Table 5. Description of sphere array

Entry	Description
$SIZE(x, y, z)$	The values of x, y and z are integer numbers that describe the size, in pixels, of the 3D image
List of pores	
$(C_x: C_y: C_z), r$	The values C_x, C_y and C_z describe the center of the pore, and the value r, its radius. The list of pores does not have limits and they can intersect

By uniformly changing the radius of the spherical pores, the porosity of the sample can be changed and with these data we approximate the linear relation of Equation (3.1.1). In the first few cases the error is less than 1.5%, except when the porosity makes the inclusions so large that they form connected shapes (Figure 18). In this case, the linear relation is not satisfied and the relative mismatch goes from 5% to 24%.

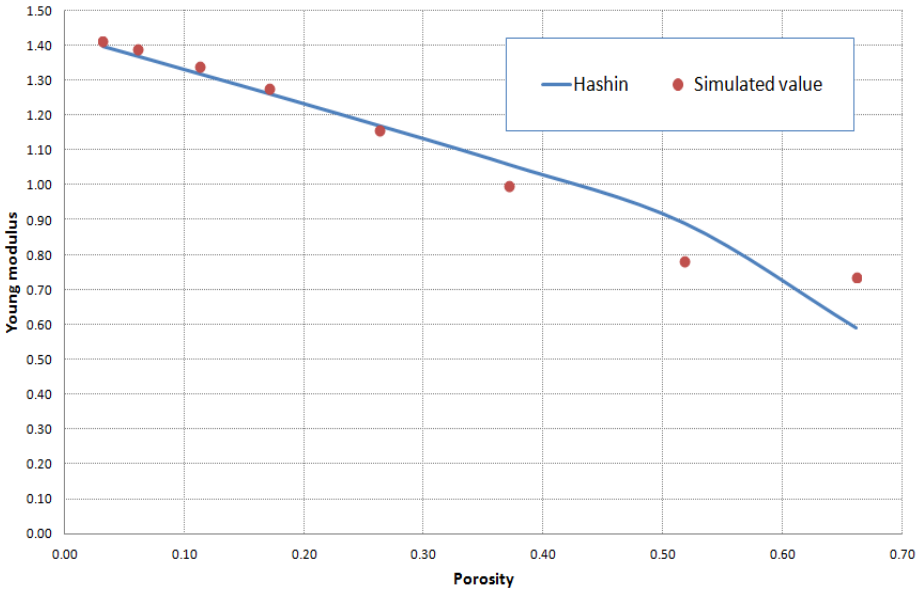

Figure 18. The simulated data is created with the Lamé constants $E_S = 1.42 \cdot 10^{10} N/m^2$ and $K_S = 3.3 \cdot 10^{10} N/m^2$ with inclusions $E_P = 0.7 \cdot 10^{10} N/m^2$ and $K_P = 1.16 \cdot 10^{10} N/m^2$. All results in the Young modulus axis must be multiplied by $10^{10} N/m^2$. The porosity oscillates from 3% to 66%. When the spherical inclusions become very large (for larger porosities), they connect each other.

3.1.3 APPROXIMATED RELATIONS

There are also boundaries for the Elastic module based on empirical approaches. Sometimes, they contradict the analytical approach and are not in direct accordance to our

FEM model [9]. The mechanical properties do not depend only on porosity, but on pore shape, location and interconnection. As we said before, there is no theoretical model that can handle all these details. On the other hand, analytical models, provide approximations that only have a range of validity [14].

Some analytical approaches use completely idealized models whose pore geometry is physically impossible in real cases, but give upper and lower boundaries, or approximations to numerically modeled or real cases [19].

An example of this, the Ramakrishnan approximation [32] makes the following assumptions:

1. The pores are spherical
2. The pores are inside spherical containers of rock solid material.
3. The ratio between pore and material rock container is the same, even though the pores and their containers can have different sizes.
4. The holes between rock containers are filled with material.

The program PIX can produce an idealized model with these characteristics. The program tries to place these pores by choosing randomly a center and a container sphere radius. If this space is free (its voxels do not correspond to any other sphere) then the sphere is placed, along with its corresponding pore (according to the ratio pore-material). If the sphere cannot be placed, the algorithm tries to find another random choice of position and radios. Depending on the number of pores specified, some of them can never be placed, so the number of tries must be limited. In our case, this limit is a linear function of the sample size. We show in Table 6 the parameters of PIX to configure spherical pore placement.

The work of Ramarkishnan validates its formula with the Finite Element Method, but with a different mesh, adapted to this kind of pore geometry. We use a voxelized approximation (Figure 19). To change the porosity we manipulate the pore radios interval, the number of pores and their sphere inclusions ratio. This produces a set of highly randomic rock samples. When only a few larges pores are placed in the rock sample, this sample is not taken into account in our experimental plot, because these few pores will not produce a certain degree of uniformity in the pore distribution (isotropy hypothesis).

Table 6. Parameters to configure pore placement

Parameter	Format	Interpretation
-rndcheese	Numeric (integer)	Total number of pores to place
-size	(N_x, N_y, N_z)	Size of the sample in pixels and enclosed between parentheses
-hollow	Numeric (float)	Specifies the ratio of the pore and the container material.
-out	String	Output filename (configuration)
-rad	(R_{min}, R_{max})	Range of pore radios (between parenthesis)

To run these tests we take the following Lamé constants: $E = 3 \cdot 10^8 \frac{N}{m^2}$ and $\upsilon = 0.21$. These could be associated to certain kind of ceramics [33]. To nullify dimensions we multiply our physical magnitudes by adequate constants. We choose:

$$C_F = N^{-1} \qquad C_L = 10^{-4}\ m$$

And set:

$$E_a = C_F C_L^2 E = 3$$
$$\upsilon = 0.21$$

So, we obtain:

$$\lambda_a = \frac{E\upsilon}{(1+\upsilon)(1-2\upsilon)} = 0.89769165$$
$$\mu_a = \frac{E}{2(1+\upsilon)} = 1.239669421$$

With these constants we construct the matrix of Equation (1.3.3). We suppose the resolution is $5\mu m$, for an image of size $50 \times 50 \times 50$. This means that the total size is $250 \cdot 10^{-6} m$. The length is set by multiplying by C_L. We get a resolution of 2.5 for the length. The rock sample size is $2.5 \times 2.5 \times 2.5$ units.

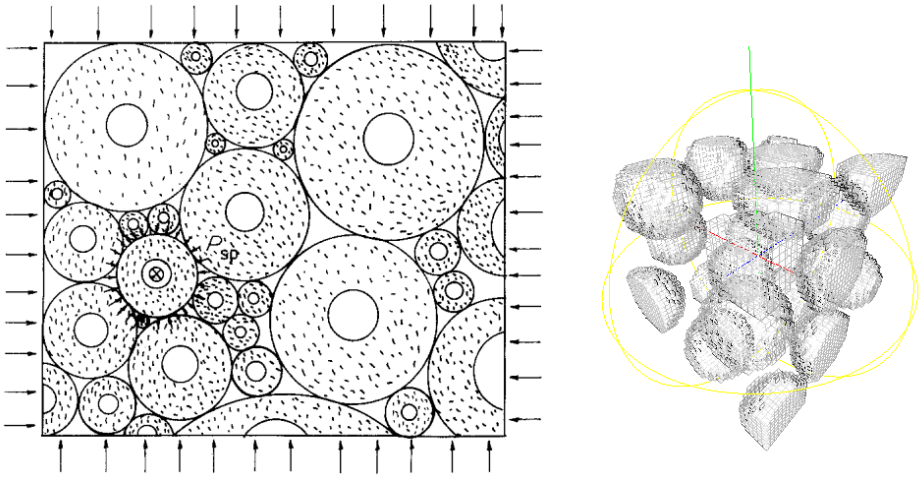

Figure 19. Theoretical pore placement and simulated pore placement in the rock sample

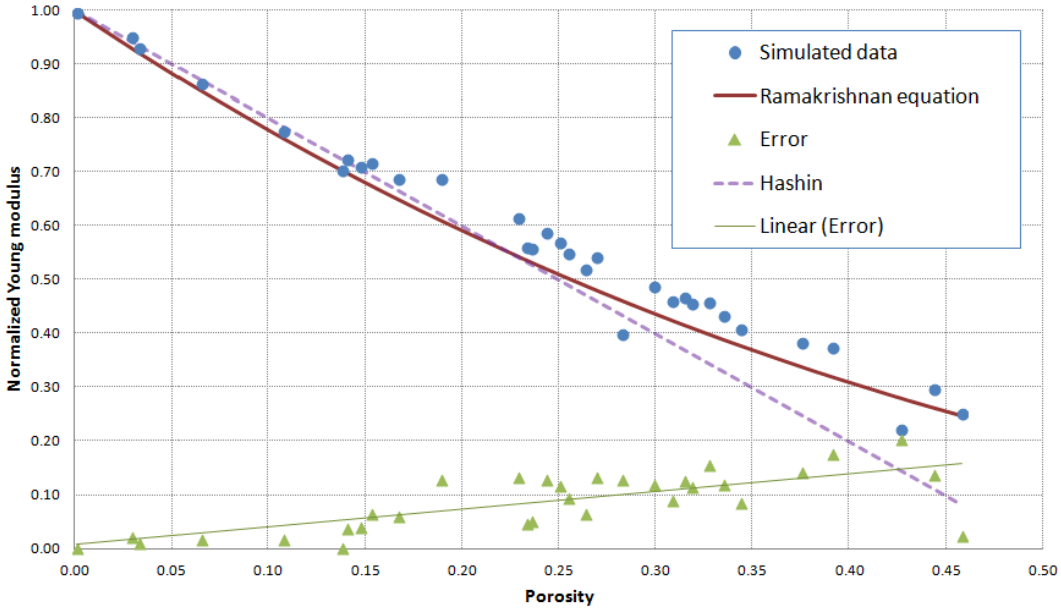

Figure 20. Comparison between Ramakrishnan approximation and our FEM simulation. The Hashin exact model applied to a porous media is also plotted for comparison purposes. The relative error between the simulated result and Ramakrishnan is plotted with its tendency line, called Linear (Error).

The results of this experiment are shown in Figure 20. For porosities of less than 45%, the relative difference between Ramakrishnan model and our simulation is less than 20% which is considered a good approximation [9]. The plotted Ramakrishnan function is:

$$\frac{E_a}{E} = \frac{(1-P)^2}{1+yP}$$

Where, according to this analysis, the value of y is contained in the interval $[2,3] \cdot v_0$. We got the best approximation with $y = 2v_0 = 0.42$. For comparison purposes, we show in this Figure 20, the linear model that Hashin proposed and we used in Section 3.1.2, setting the elastic constants for the pores to zero. This model offers a very good approximation for low porosities, with a relative mismatch of less than 10%. After approximately 25% porosity, it does not capture the rock sample behavior (even with this idealized model), producing larger relative errors.

One of the most used approximations is given by Hashin-Shtrikman [19,9,30]. For biphasic materials the upper and lower bounds for the bulk and Poisson modulus are given by:

$$K^{HS} = K_1 + \frac{f_2}{(K_2 - K_1)^{-1} + f_1\left(K_1 + \frac{4}{3}\mu_1\right)^{-1}}$$

$$\mu^{HS} = \mu_1 + \frac{f_2}{(\mu_2 - \mu_1)^{-1} + 2f_1(K_1 + 2\mu_1)\left[5\mu_1\left(K_1 + \frac{4}{3}\mu_1\right)\right]^{-1}}$$

Where the index 1 denotes the Lame constants of the matrix, and the index 2, denotes the Lame constants of the voids. The values of f_1 and f_2 denote the respective volume proportion. By interchanging the indices 1 and 2, we get the upper and lower limits. To calculate the Young modulus we use the following relation:

$$E = \frac{9K\mu}{3K + \mu}$$

In which the values of K and μ are chosen to maximize or minimize the value of E. This procedure can be plotted and compared with random porous rock samples, generated with the same method we illustrated in Figure 19. The result is shown in Figure 21. We also show the linear model of Section 3.1.2 for comparison purposes. This linear model is adequate only for low porosities, and the random samples Young modules are always contained within the Hashin-Shtrikman limits. This figure is obtained with the following mechanical properties:

$$\lambda_S = 0.89769150 \quad \mu_S = 1.239669421$$

And:

$$\lambda_P = \frac{\lambda_S}{8} \qquad \mu_P = \frac{\mu_S}{8}$$

We use the same normalizer constants $C_F = N^{-1}$ and $C_L = 10^{-4}m$. This satisfies Hashin hypothesis according to which the mechanical constants do not differ by a proportion larger than 10.

The Reuss and Hashin lower limit become zero when one set of Lame constants is zero. This means that these limits are very wide for porous media.

Hashin-Shtrikman limits are not adequate for porous rocks, in which pores are modeled as zero constant elements. The lower limit becomes zero, giving a very wide error interval.

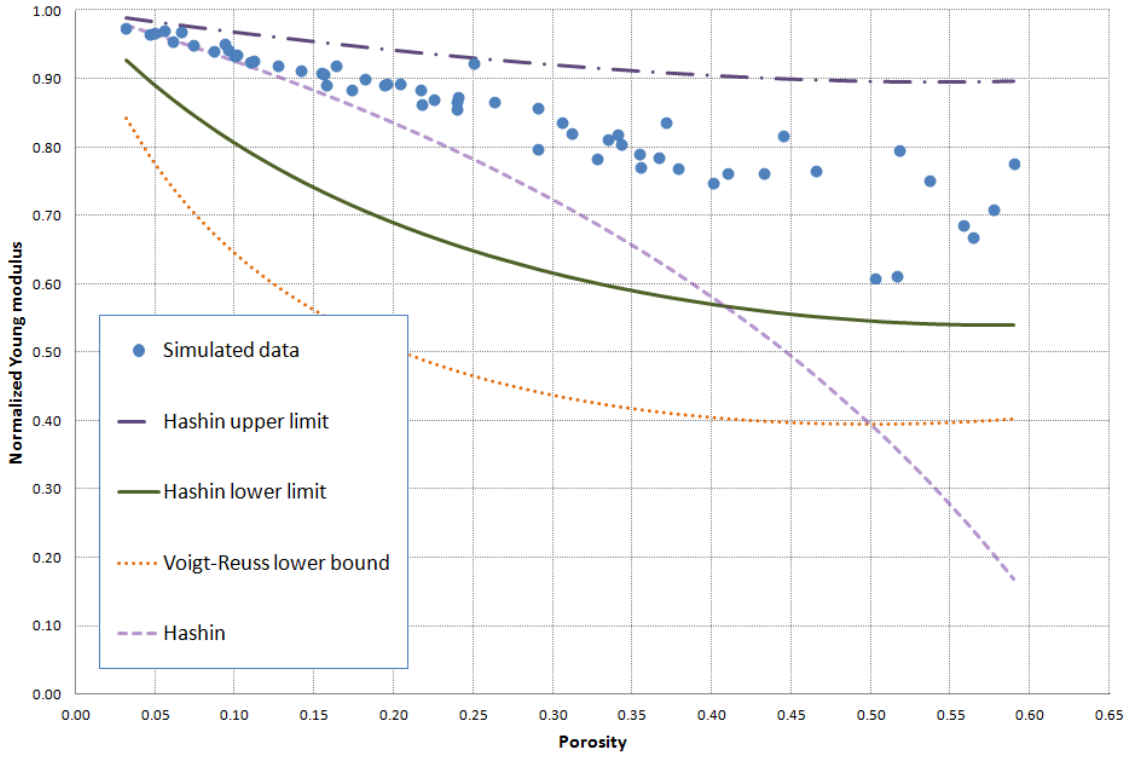

Figure 21. The Hashin-Strikhman upper and lower limits are displayed along with our simulated results. The Voigt-Reuss inferior bound is displayed. All data are normalized with the upper Voigt-Reuss limit. Hashin linear approximation is also displayed.

3.2 SANDSTONE ELASTIC MODULUS

3.2.1 CASE STUDY

In this section and forward we will consider a real example of a sandstone. We will suppose it is biphasic with the following mechanical properties

$$E = 50 \cdot 10^9 N/m^2$$
$$\upsilon = 0.25$$

We nullify the dimensions by setting:

$$C_F = N^{-1}$$
$$C_L = 10^{-5} m$$

Getting:

$$E_a = C_F C_L^2 E = 5$$
$$\upsilon_a = 0.25$$

The Lamé constants that are useful for the local stiffness matrix are given by:

$$\lambda = \frac{E_a \upsilon_a}{(1+\upsilon_a)(1-2\upsilon_a)} = 2$$
$$\mu = \frac{E_a}{2(1+\upsilon_a)} = 2$$

The rock sample size is $8.683\mu m$ per voxel, and a resolution of $300 \times 300 \times 300$. The total size of the rock sample is: $2604.9\mu m \times 2604.9\mu m \times 2604.9\mu m$.

We nullify the dimensions of these lengths by setting:

$$l_a = C_L^{-1} l = 260.49$$

With a resolution of 0.8683 units per voxel length. Our case will be the Blunt image S1.dat. The unconfined compression test for this image, whose porosity is 0.141303 for a displacement of -12 units gives the result, for the elastic modulus of the rock sample:

$E_e = 3.187779196$ units. That is $E_e = 3.18 \ldots \cdot 10^{10} N/m^2$. This takes about 4 hours of CPU time. The finite element mesh contains more than 23 millions of elements with almost 70 millions degrees of freedom. This is a large number of unknown variables for the conjugate gradient to solve.

Let us consider a different approach. We divide the rock sample in 8 sections of the same volume. The size of each section is $150 \times 150 \times 150$ voxels. We show in Table 7 the porosity and Young modulus in each one of these rock sample sections. These results were taken by applying an unconfined compression test in each subsample with a -12 units displacement.

Table 7. Young modulus for sample's different spatial sections

Front Section		Back section	
Porosity: 0.137 Young Modulus: 3.363	Porosity: 0.161 Young Modulus: 3.047	Porosity: 0.150 Young Modulus: 3.125	Porosity: 0.154 Young Modulus: 3.017
Porosity: 0.129 Young Modulus: 3.512	Porosity: 0.134 Young Modulus: 3.183	Porosity: 0.130 Young Modulus: 3.457	Porosity: 0.135 Young Modulus: 3.335

Each rock section has similar pore characteristics respect the entire sample, giving a similar Young module. The relative difference in the elasticity modulus of each area is always less than 16%. In these kinds of tests, this is considered a good approximation [9]. We can say that this sample has a high degree of homogeneity, and subsamples of size 150^3 capture it adequately. Solving this resolutions takes about 30 minutes of CPU time. Considering these results, we take a subsample of the image, centered at the voxel position $(150,150,150)$ and size $200 \times 200 \times 200$. The subsample volume is $173.66 \times 173.66 \times 173.66$ cubic units. We show a sequence of four images in Figure 22 . Once again, an upper displacement of -12 units is applied.

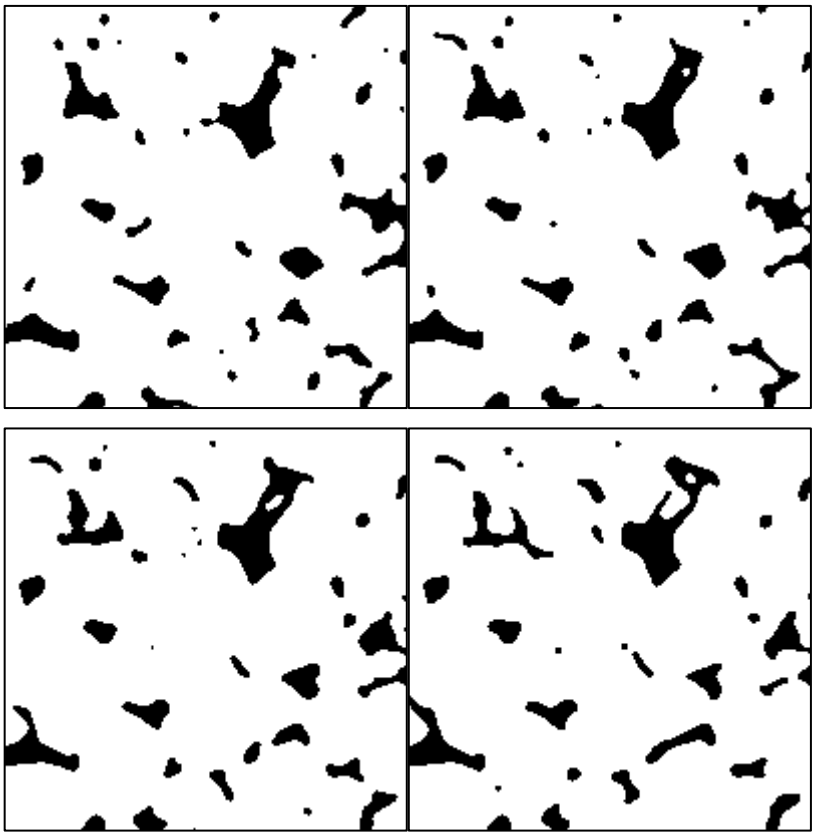

Figure 22. A sequence of sandstone S1 images. These are the first four images of the sequence.

This rock sample produces a linear system with 21 886 790 unknown variables. The porosity of the subsample is 0.1400671 and the Young modulus we obtain, after 45 minutes of CPU time, is $E = 3.316812$ units.

We show in Figure 23, the convergence curve of the integrated force according to the number of iterations. The stop criterion is a relative difference of 0.05% between these values.

The 3 dimension problem scales very quickly. A relatively small system $200 \times 200 \times 200$ contains about $2 \cdot 10^7$ unknown variables, but a $300 \times 300 \times 300$ problem contains almost an additional order of magnitude, $9 \cdot 10^7$ unknown variables. This issue makes necessary the use of the "element by element" approach we show in Section 2.4, in most workstations.

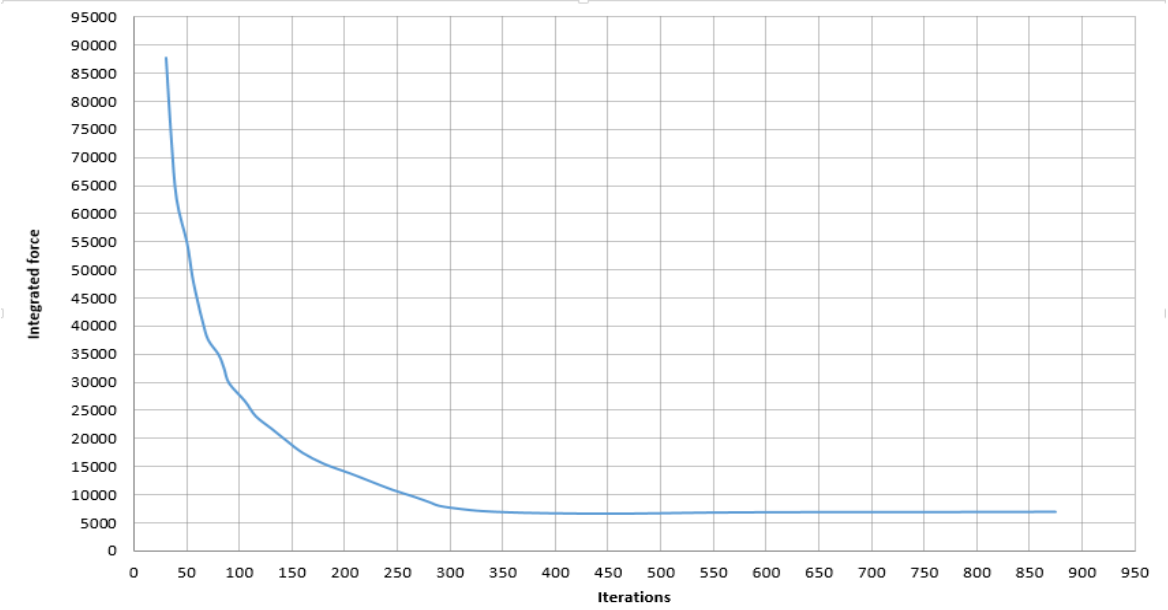

Figure 23. Integrated force (in N) as function of the number of iterations.

3.2.2 MESH SIMPLIFICATION

We can try to build a three dimension mesh with fewer elements, capturing the actual pore geometry. This can be done with some variations of Voronoi algorithms, placing nodes in key positions [1]. Another approach is to use simple digital image processing algorithms [16], to change input image resolution, and maintaining the same policy we have used so far (each voxel is an element).

We take the input image of the last example. The large size of the pores (Figure 22) may allow the definition of a larger voxel size, producing less precision in pore representation. We use a 3D image of size $100 \times 100 \times 100$, obtained by a reduction interpolation method. This means that the rock sample size will be the same, but in this case, each voxel will have a length of 1.7366 units. To calculate the voxel color (x, y) in the smaller image, we use the following procedure:

1. Calculate the color average a_v of the box enclosed by voxels $(x, y) \longrightarrow (2x, 2y)$ in the original image.
2. Define (x, y) as pore if $a_v < a_0$ and as rock material if $a_v \geq a_0$. The value of a_0 is chosen in such a way that the new image porosity be approximately the same of the original image.

In this case, we choose $a_0 = 135$ for a porosity of 0.148126. The relative difference between the larger image porosity and its reduction is 5%. We show in Figure 24 some of these new smaller images. With this new size, the memory space of four images in the larger image is now occupied by 8 images. This is a significant reduction. The new system size is of 2 848 698 unknown variables and it is solved in 3 minutes of CPU time.

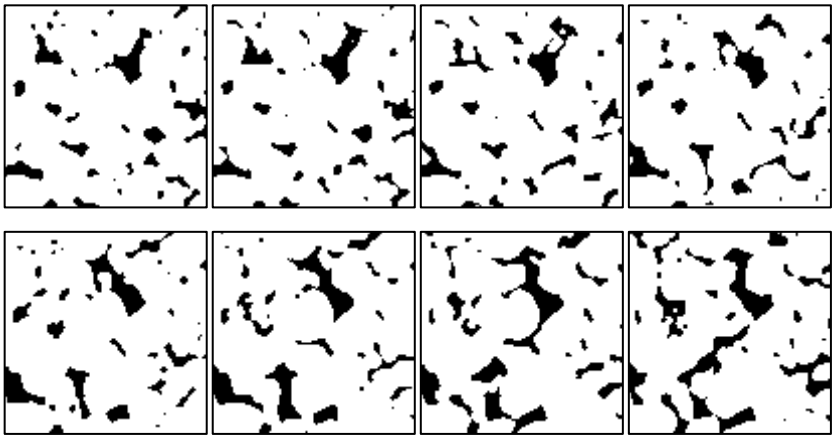

Figure 24. The first 8 images of sandstone S1, with less resolution. The image size is $100 \times 100 \times 100$, which means that the memory space of 4 images can now be occupied by 8 images.

With this new model, the Young modulus we get is $E = 3.268048 C_L^{-2} C_F^{-1}$. Despite the huge simplification we just described, the relative difference between the first and second resolution is only 1.5%.

We proceed to make another simplification, reducing the original mesh to $50 \times 50 \times 50$ voxels. In this case we get a Young modulus $E = 3.510341 C_L^{-2} C_F^{-1}$. The relative difference between the original sample resolution and this is less than 6%. We can say that the image resolution has lost much precision , but even with this enormous simplification, the difference is not very dramatic.

As a final test, we resize the image to $20 \times 20 \times 20$. We get $E = 3.917537 C_L^{-2} C_F^{-1}$ with a relative difference of 15%. We can conclude that this reduction technique shows a relationship between pore geometry and Young modulus, without considering porosity. Each reduction sample has the same approximate porosity of the original rock sample, but the reduction produces unavoidable pore geometry changes (which are more important if the pore contains only a few voxels).

The resizing factor must not always be an integer. We can resize the original image to any size using simple linear interpolation [16]. We show in Figure 25 the convergence process to a the fixed value according to the image resolution. As shown in this figure, in all cases the porosity remains practically the same. The image resolution is different in each point, as well as the precision with which the pore geometry is captured by this resolution.

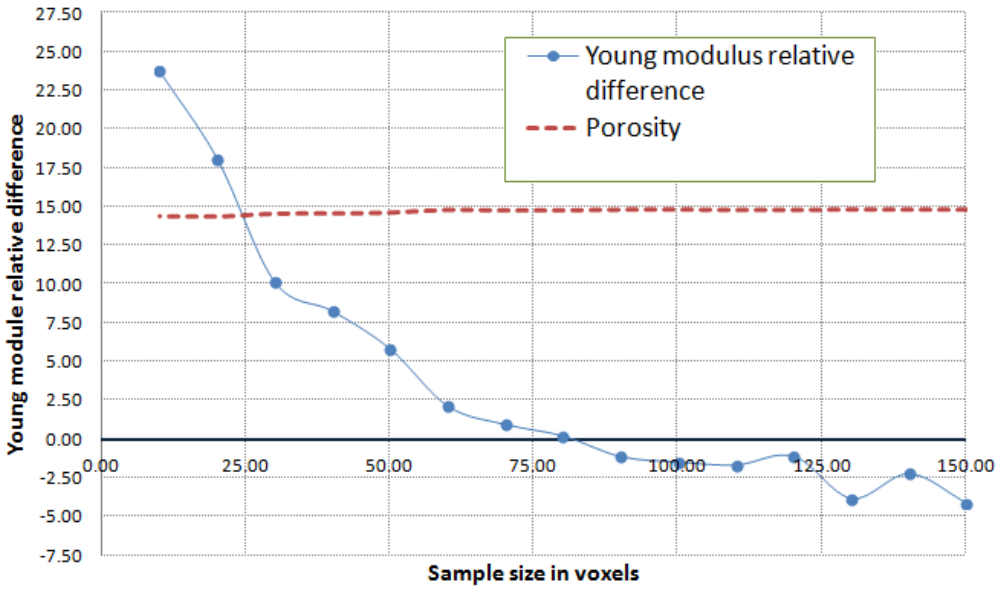

Figure 25. Estimated Young modulus error with respect the original size and in different resolutions. The almost constant porosity is displayed.

From this plot we can make the following conclusions:

1. Even using large fractional resizing factors (for example $0.75 = \frac{3}{4}$ for the size $150 \times 150 \times 150$), the linear interpolation of fraction resizing seems to induce an important error, comparable to the $0.25 = \frac{1}{4}$ factor reduction of the sample $50 \times 50 \times 50$.
2. The element by voxel philosophy can be dramatically simplified, without producing dramatic changes in mechanical properties estimation.
3. Porosity is still the most important factor in mechanical parameters estimation. Even with large simplifications (for example $\frac{1}{20}$ for a size $10 \times 10 \times 10$) the relative difference with respect the largest resolution is less than 25%.

We can use computer graphic techniques (variations of Delaunay Triangulations in three dimensions [1]) to create a simplified mesh and capture the essential pore shape. This will produce a computationally feasible problem and is a path for new research.

3.2.3 CHANGING ROCK POROSITY

Classical digital image processing algorithms such as erosion and dilation can be used to change rock sample porosity without significantly alter pore geometry. In its most simplistic form, these algorithms can be applied in each image of the rock sample [16]. We show in Figure 27 consecutive erosions and two consecutive dilations of the first image of the rock sample. This kind of sample manipulation allows us to get a curve for the Young modulus and not only one approximation. As opposed to the last case, here we are changing rock porosity and the pore geometry remains essentially the same. We apply the same mechanical displacement we applied in Section 3.2.1, and the image resolution remains the same: $200 \times 200 \times 200$ in all cases. Two dilations and two erosions are applied.

We show in Figure 26 the simulated results, compared with the Ramakrishnan estimation (using $y = 3\nu_0$). In all cases the relative difference is below 6%. It is important to remark that Ramakrishnan approximation depends only on porosity and not on pore geometry.

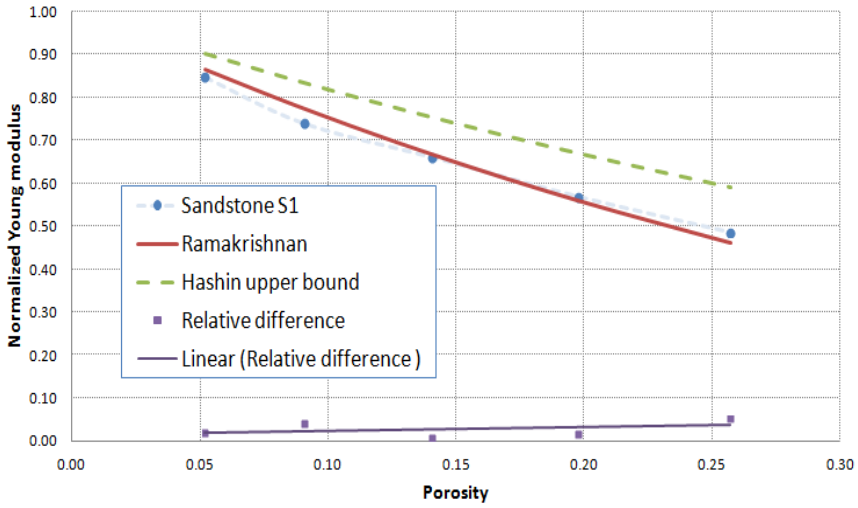

Figure 26. Porosity change in Sandstone case study. The simulated results are compared with Ramakrishnan estimate and Hashin upper limit for porous media. The displayed relative difference is with respect to Ramakrishnan model. It is almost constant.

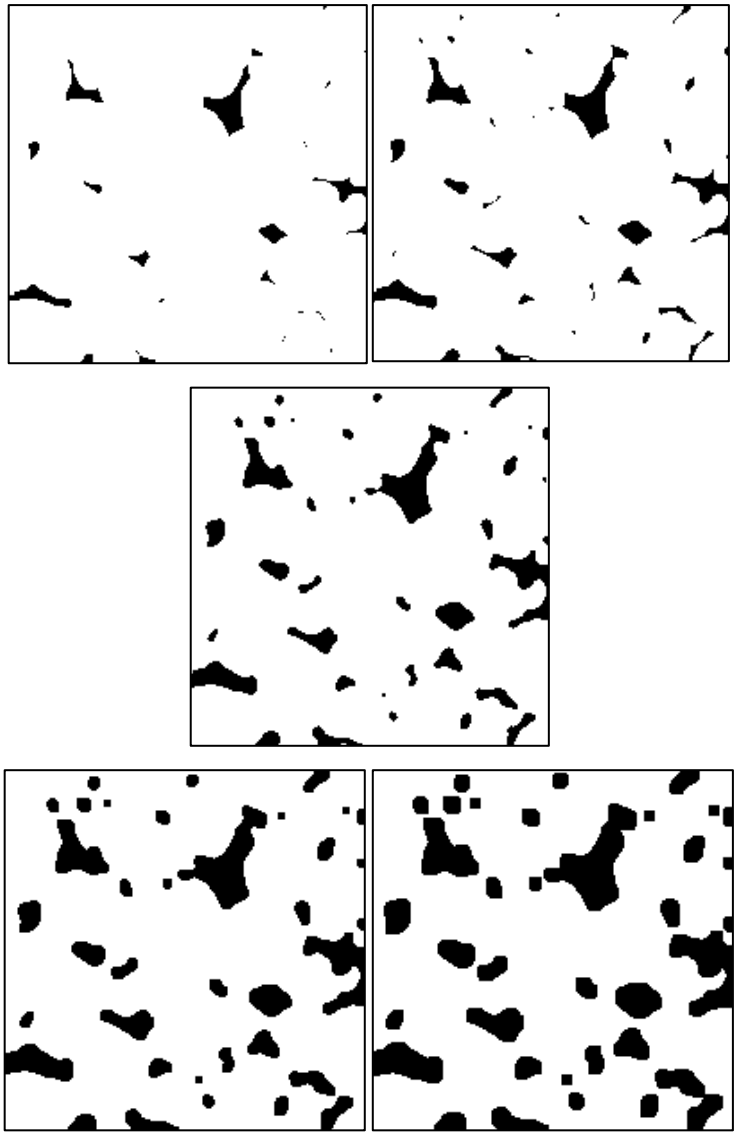

Figure 27. Sandstone dilation and erosion. The kernel in all cases is a simple square. Two successive dilations and two successive erosions are applied.

We conclude that the most determinant aspect of the Young module is porosity. This gives us the general path to a smaller, more computationally feasible mesh for mechanical simulations. If the rock sample pores are big enough, the mesh can be simplified dramatically. This simplification should not alter the porosity. The pore geometry can be approximated with less precision, using simple averages. Determining an adequate mesh has a direct incidence on the technique applicability, as a large number of unnecessary elements consume a lot of CPU time.

3.2.4 ANISOTROPY

We cannot say that any rock sample is completely anisotropic, but we can try to measure the level of uniformity in pore distribution, using mechanical tests. We propose to apply a variation of the unconfined compression at different rotation angles of the rock sample. As the sample is cubic, this is physically impossible. Therefore, we cut a cylindrical piece of the cubic sample rock. This can be easily rotated across the transversal axis (axis Z) at several angles. The force is applied along the upper part cylinder perimeter, applying a specified diameter contraction. This is similar to Brazilian test, in which the applied linear model is unable to detect rock fracture [27]. We show in Figure 28 the three dimension shape of the rock sample.

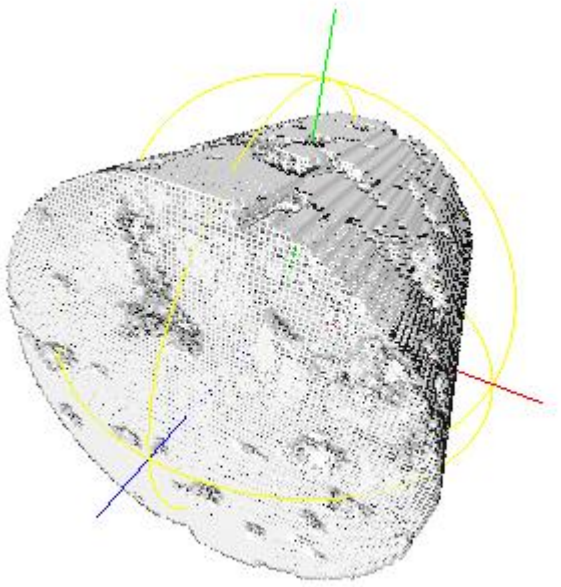

Figure 28. Cylindrical shape rock sample

The rotation is achieved by applying classical digital image processing techniques in each image of the sample. We apply the shear rotation method to rotate the image sample at angles of 45 degrees multiples, suppressing all information outside the cylinder [41]. The rotation can be applied to the mesh model, as well, but applying image rotation it is easier to establish border conditions, using the natural error induced by the voxel discretization.

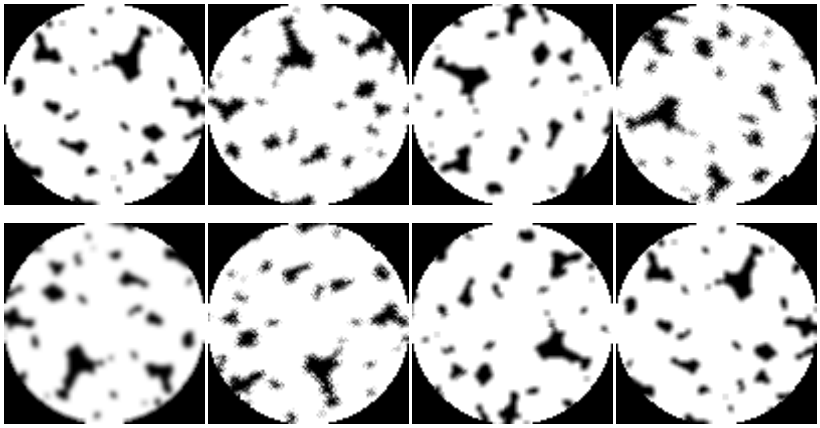

Figure 29. Successive image rotations in 45 degrees multiples

We show in Figure 29 the different rotation angles applied to the first image of the rock sample. The sample size is $100 \times 100 \times 100$. Despite the geometrical conditions change, the force necessary to contract the cylinder diameter in 12 units does not change in more than 17%. The chaotic behavior of the force for the different angles is shown in Figure 30.

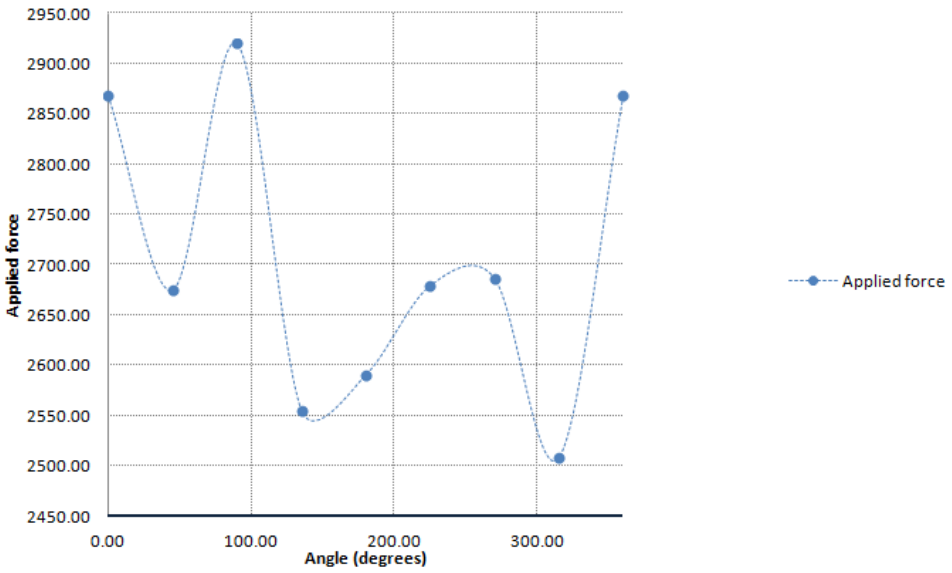

Figure 30. Force applied to the cylindrical sample in function of angles

The complementary value (approximately 83%) can be considered a measure of the sample anisotropy degree. The exact influence of this factor in the rock mechanical properties is an open problem and a future research path.

CONCLUSIONS

This works represents a starting point in implementing computer software tools for digital petrophysics. All developed tools are classical, and they do not represent a novelty in the finite element method or in the simulation techniques. Nevertheless, our development applies these classical methods in the particular context of the microtomographic images and elastic modulus estimation.

One of our first choices is to define one finite element per voxel (or pixel) and four adjacent nodes in each one of them. This seems to be a common choice in most implementations, because of its simplicity and precision: the finite element method mesh that is created with this mechanism has the same precision of the image acquiring device. But as we have shown in Chapter 2, this choice has a drawback in terms of performance. The other alternative we described (mounting an explicit matrix with a sparse structure) requires a lot of memory to represent the sparse matrix (even using adequate data structures). But it is considerably faster. This second alternative is useful for small meshes. But, as we have shown in Chapter 3, the element per voxel policy can be considerably simplified. This defines a research path: a mechanism to create a mesh that captures pore geometry and rock porosity, simple enough to be computationally feasible, solvable in a few seconds. This would not be a symmetric mesh, and the element by element technique is not applicable, or could be extremely inefficient. In this case, the large stiffness matrix must be assembled. Another consideration here is that the microtomographic image should capture pore geometry with a good resolution. A pore characterized by only a few voxels is not suitable to be simplified nor should be considered a good approximation of the physical pore.

Our proposed workflow begins with a sequence of microtomographic images and ends with a mechanical Lame constant: the Young modulus. These images are not necessarily static and can be submitted to different processes. In CHAPTER 3, we have shown two of them: image reduction and porosity manipulation. This kind of image manipulations allows us to isolate two key concepts in digital rock physics: porosity (the ratio between pore and solid material) and pore shape. The separation of these two properties is only possible in a simulated computer environment. It can't be done physically.

We believe that finding a method to create a simplified finite element mesh that captures adequately pore geometry while preserving its porosity is a way to understand the influence of pore geometry and borders in the mechanical parameters estimation. An open problem is to determine if the voxelized discretization is the best mesh representation for the finite element method or if it can be modified by purely geometrical techniques in order to get better approximations of pore contours. These kinds of comparisons should be performed with real laboratory data.

There are also some technological issues. We have created an implementation of the FEM method, applicable to Digital Rock Physics in CHAPTER 2. This implementation leads to a deeper understanding of the different steps in digital rock physics and the different choices we can make. One of them is not to use periodic boundary conditions. This is assumed in almost all implementations, but we are not using them. They can't be reproduced physically in mechanical laboratory experiments. This forces us to use large rock samples, that capture the general rock pore distribution. A future research path is to distinguish this difference and determine precisely the need of periodic boundary conditions in certain cases.

Another technological choice is the way all numerical methods were parallelized. This was done at CPU level, using SIMD and MP parallelization. We did not note any significant gain in simply passing some of these processes (vector addition, product, matrix operations, etc) to GPU. Also, the large size of our models limits the GPU applicability. But the most efficient alternative should not be to use one technique or another, but both, complementing each other according to their efficiency in each specific step. This should improve our implementation performance.

The methodology we have illustrated in CHAPTER 3 uses digital image processing techniques to obtain more information from the rock sample images. With simple erosion and dilation techniques, we can get a curve of the Young modulus estimate as a function of porosity. This allows the comparison of several Elastic Modulus values for different porosities, with analytical estimates. Determining the estimation curve parameters that better adjust to simulated data, can lead to a better understanding of rock elastic properties. Another Digital Image Processing technique is the image rotation, manipulating the geometry of how the uniaxial test is applied. This allows a measure of the level of pore isotropy in the rock

sample. This level of isotropy depends on the test being applied. As far as we know, these kinds of techniques have not been used in Digital Rock Physics.

The future research paths we have described are natural consequences of our current state. But there is no doubt that this topic has a lot of interesting and useful trends that can be applied in different contexts.

BIBLIOGRAPHY

[1] Max K. Agoston, *Computer graphics and geometric modeling; Implementation and algorithms*, Springer-Verlag London, Ed.: Springer, 2004.

[2] Max K. Agoston, *Computer graphics and geometric modeling; Mathematics*, Springer-Verlag London, Ed.: Springer, 2005.

[3] Heiko Andrä et al., "Digital Rock Physics benchmarks-Part I: Imaging and Segmentation," *Comput. Geosci.*, vol. 50, pp. 25-32, #jan# 2013. [Online]. http://dx.doi.org/10.1016/j.cageo.2012.09.005

[4] Heiko Andrä et al., "Digital Rock Physics Benchmarks-part II: Computing Effective Properties," *Comput. Geosci.*, vol. 50, pp. 33-43, #jan# 2013. [Online]. http://dx.doi.org/10.1016/j.cageo.2012.09.008

[5] C. H. Arns, M. A. Knackstedt, W. V. Pinczewski, and E. J. Garboczi, "Computation of linear elastic properties from microtomographic images: Methodology and agreement between theory and experiment," *Geophysics*, vol. 67, p. 1396, 2002.

[6] RandolphE. Bank and CraigC. Douglas, "Sparse matrix multiplication package (SMMP)," *Advances in Computational Mathematics*, vol. 1, no. 1, pp. 127-137, 1993. [Online]. http://dx.doi.org/10.1007/BF02070824

[7] Martin J. Blunt et al., "Pore-scale imaging and modelling," *Advances in Water Resources* , vol. 51, no. 0, pp. 197-216, 2013, 35th Year Anniversary Issue. [Online]. http://www.sciencedirect.com/science/article/pii/S0309170812000528

[8] Barbara Chapman, Gabriele Jost, and Ruud van der Pas, *Using OpenMP: Portable Shared Memory Parallel Programming (Scientific and Engineering Computation)*.: The MIT Press, 2007.

[9] J.A. Choren, S.M. Heinrich, and M.B. Silver-Thorn, "Young modulus and volume porosity relationships for additive manufacturing applications," *Journal of Materials Science*, vol. 48, no. 15, pp. 5103-5112, 2013. [Online]. http://dx.doi.org/10.1007/s10853-013-7237-5

[10] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to Algorithms, Third Edition*, 3rd ed.: The MIT Press, 2009.

[11] Jane M. Dewey, "The Elastic Constants of Materials Loaded with Non x2010;Rigid Fillers," *Journal of Applied Physics*, vol. 18, no. 6, pp. 578-581, Jun 1947.

[12] E. Fjar, R.M. Holt, A.M. Raaen, R. Risnes, and P. Horsrud, *Petroleum Related Rock Mechanics: 2nd Edition.*: Elsevier Science, 2008. [Online]. http://books.google.com.br/books?id=l6CfasFxhzYC

[13] David C. Fong and Michael Saunders, "CG versus MINRES: An empirical comparison," Stanford University, Tech. rep. 2011. [Online]. http://www.stanford.edu/group/SOL/reports/SOL-2011-2R.pdf

[14] E.J. Garboczi and J.G. Berryman, "Elastic moduli of a material containing composite inclusions: effective medium theory and finite element computations ," *Mechanics of Materials* , vol. 33, no. 8, pp. 455-470, 2001. [Online]. http://www.sciencedirect.com/science/article/pii/S0167663601000679

[15] Gene H. Golub and Charles F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press;, Ed.: The Johns Hopkins University Press; 3rd edition, 1996.

[16] Rafael C. and Woods, Richard E. Gonzalez, *Digital Image Processing, Third Edition*, Prentice Hall, Ed.: Pearson, 2008.

[17] J.P. Harrison and J.A. Hudson, *Engineering rock mechanics - An introduction to the principles.*: Elsevier Science, 2000. [Online]. http://books.google.com.br/books?id=GNoTr0T84NYC

[18] Z. Hashin, "Analysis of Composite Materials---A Survey," *Journal of Applied Mechanics-transactions of The Asme*, vol. 50, 1983.

[19] Z. Hashin and S. Shtrikman, "A variational approach to the theory of the elastic behaviour of multiphase materials ," *Journal of the Mechanics and Physics of Solids* , vol. 11, no. 2, pp. 127-140, 1963. [Online]. http://www.sciencedirect.com/science/article/pii/0022509663900607

[20] Z. Hashin and S. Shtrikman, "A variational approach to the theory of the elastic behaviour of polycrystals ," *Journal of the Mechanics and Physics of Solids* , vol. 10, no. 4, pp. 343-352, 1962. [Online]. http://www.sciencedirect.com/science/article/pii/0022509662900054

[21] Z. Hashin and S. Shtrikman, "On some variational principles in anisotropic and nonhomogeneous elasticity ," *Journal of the Mechanics and Physics of Solids* , vol. 10, no. 4, pp. 335-342, 1962. [Online]. http://www.sciencedirect.com/science/article/pii/0022509662900042

[22] R. Hill, "Elastic properties of reinforced solids: Some theoretical principles ," *Journal of the Mechanics and Physics of Solids* , vol. 11, no. 5, pp. 357-372, 1963. [Online]. http://www.sciencedirect.com/science/article/pii/002250966390036X

[23] "http://www.noticias.uff.br/noticias/2012/12/assina-meio-ambiente-e-energia.php".

[24] T.J.R. Hughes, *The finite element method: linear static and dynamic finite element analysis*.: Dover Publications, 2000. [Online]. http://books.google.com.br/books?id=yarmSc7ULRsC

[25] O. Ishai and L.J. Cohen, "Elastic properties of filled and porous epoxy composites ," *International Journal of Mechanical Sciences* , vol. 9, no. 8, pp. 539-546, 1967. [Online]. http://www.sciencedirect.com/science/article/pii/0020740367900537

[26] International Organization for Standardization ISO, "ISO/IEC/IEEE 60559:2011 Information technology -- Microprocessor Systems -- Floating-Point arithmetic. ICS: 35.160. Stage: 60.60 (2011-07-07). TC/SC: ISO/IEC JTC 1/SC 25. Number of Pages: 58," *ISO Standards catalogue*, 2011.

[27] Ye Jianhong, F.Q. Wu, and J.Z. Sun, "Estimation of the tensile elastic modulus using Brazilian disc by applying diametrically opposed concentrated loads ," *International Journal of Rock Mechanics and Mining Sciences* , vol. 46, no. 3, pp. 568-576, 2009. [Online]. http://www.sciencedirect.com/science/article/pii/S1365160908001366

[28] Mark A. Knackstedt et al., "Digital rock physics: 3D imaging of core material and correlations to acoustic and flow properties," *The Leading Edge*, vol. 28, no. 1, pp. 28-33, 2009. [Online]. http://tle.geoscienceworld.org/content/28/1/28.abstract

[29] Rainer Kress, *Linear Integral Equations. Second Edition*, Springer-Verlag New York, Ed.: Springer, 1999.

[30] Gary Mavko, Tapan Mukerji, and Jack Dvorkin, *The Rock Physics Handbook: Tools for Seismic Analysis of Porous Media*, 2nd ed. Cambridge: Cambridge University Press, #may# 2009. [Online]. http://www.worldcat.org/isbn/0521861365

[31] F. Mees and Geological Society of London, *Applications of X-ray Computed Tomography in the Geosciences*.: Geological Society, 2003. [Online]. http://books.google.com.br/books?id=AuKNi2UYSxIC

[32] N. Ramakrishnan and V.S. Arunachalam, "Effective elastic moduli of porous solids," *Journal of Materials Science*, vol. 25, no. 9, pp. 3930-3937, 1990. [Online]. http://dx.doi.org/10.1007/BF00582462

[33] R.W. Rice, *Mechanical Properties of Ceramics and Composites: Grain And Particle Effects*.: Taylor \& Francis, 2000. [Online]. http://books.google.com.br/books?id=vl7gLP0bnwsC

[34] R. Richards, *Principles of Solid Mechanics*.: Taylor \& Francis, 2000. [Online]. http://books.google.com.br/books?id=HlSBWwHuCK0C

[35] A. P. Roberts and E. J. Garboczi, "Computation of the linear elastic properties of random porous materials with a wide variety of microstructure," *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 458, no. 2021, pp. 1033-1054, 2002. [Online]. http://rspa.royalsocietypublishing.org/content/458/2021/1033.abstract

[36] L.A. Schoof and V.R. Yarberry, *EXODUS II: A finite element data model.*, Sep 1994. [Online]. http://www.osti.gov/scitech/servlets/purl/10102115

[37] W.S. Slaughter, *The linearized theory of elasticity*.: Birkhauser, 2002. [Online]. http://books.google.com.br/books?id=a_8_AQAAIAAJ

[38] The OpenMP API Specification for parallel programming, *OpenMP C and C++ Application Program Interface*, Downloadable from: http://openmp.org/wp/openmp-specifications/, Ed.: OpenMP Architecture Review Board, 2002.

[39] Djebbar Tiab and Erle C Donaldson, *Petrophysics: Theory And Practice Of Measuring Reservoir Rock And Fluid Transport Properties*. Boston: Gulf Professional Pub., 2012. [Online]. http://isbnplus.org/9780123838483

[40] Lloyd N. Trefethen and David Bau, *Numerical Linear Algebra*.: SIAM, 1997.

[41] M. Unser, P. Thevenaz, and L. Yaroslavsky, "Convolution-based interpolation for fast, high-quality rotation of images," *Image Processing, IEEE Transactions on*, vol. 4, no. 10, pp. 1371-1381, oct 1995.

[42] B. van Rietbergen, H. Weinans, R. Huiskes, and A. Odgaard, "A new method to determine trabecular bone elastic properties and loading using micromechanical finite-element models ," *Journal of Biomechanics* , vol. 28, no. 1, pp. 69-81, 1995. [Online]. http://www.sciencedirect.com/science/article/pii/0021929095800085