DAMIAN-PAR: A Numerical Tool for the Simulation of Wave Propagation Problems Over Large Scale Seismic Scenarios Based Upon the Finite Element Method

Ricardo Serrano Salazar
Escuela de Ingeniería
Universidad EAFIT
Medellín
Colombia
rserrano@eafit.edu.co

May 12, 2014

DAMIAN-PAR: A Numerical Tool for the Simulation of Wave Propagation Problems Over Large Scale Seismic Scenarios Based Upon the Finite Element Method

Ricardo Serrano Salazar

A thesis presented for the degree of
Master of Science

Escuela de Ingeniería
Universidad EAFIT
Colombia
May 12, 2014

Chapter 1

Introduction

The simulation of earthquakes is necessary to have an understanding about how seismic waves that propagate through earth can affect urban areas sometimes producing catastrophic losses at the surface. Usually, researchers model simplified earthquakes to have a closer understanding of the phenomena; however, realistic simulations of actual earth models and earthquake sources are necessary for an understanding that can be applied to the prevention and management of disasters at a vulnerable region. Realistic simulations of earthquakes provide challenges that are not foreseen in their simplified counterparts, like the modeling of a region that comprises hundreds of kilo-meters and whose materials might have non-linearities or at least be highly heterogeneous. The model can also include a seismic source modeled with high frequencies. The detailed simulation of a model with these complexities implies large computational demands exceeding the standard capabilities of practicing engineers. In other words, to model such problem it's necessary to develop software that takes advantage of a lot of computational resources in the form of a computer cluster. A computer cluster is a set of computer servers connected through a fast network that can run a single program in all of them at the same time. We call simulations that require and use such computing resources *large scale simulations*.

Large scale simulations have been used to reproduce ground motions induced by earthquakes in regions of important seismic activity. For example, Southern California is very vulnerable to earthquakes and researchers usually try to simulate historical earthquakes that occurred in the region using thousands of computing nodes to reproduce the seismograms or to obtain information about the earth's properties departing from those earthquakes.

Universidad EAFIT, located in Medellín which is Colombia's second largest city, is trying to push projects that make use of APOLO, a new computing cluster with hundreds of computing nodes that is intended for research in several of EAFIT's key fields. Grupo de Mecanica Aplicada at EAFIT has been working for years solving wave propagation problems and it has seen the acquisition of APOLO as an opportunity to enhance its capabilities and try to appropriate large scale simulation technologies to solve problems with realistic sizes in Aburra's Valley. The challenges involved from the software development standpoint had kept Grupo de Mecanica Aplicada from doing so.

The purpose of this Master of Science project is to implement a numerical tool to conduct large scale simulations of plane waves propagating over realistic seismic scenarios in the cluster APOLO, the high performance computing facility at Universidad EAFIT. The resulting numerical tool would be the first of its kind in the region and will allow Grupo de Mecanica Aplicada to continue working

in the study of topographic and local site effects in earthquake engineering. The specific tool developed in this work is the 2D version of HERCULES, the 3D end-to-end earthquake simulation tool implemented at Carnegie Mellon University by the Mechanics, Materials and Computing group led by Professor Jacobo Bielak. The 2D code, implemented in this work is part of DAMIAN, the family of numerical simulation programs developed by Grupo de Mecanica Aplicada for the study of wave propagation phenomena in different contexts. As such, this specific numerical tool has been termed DAMIAN-PAR. This report describes he main contributions of the current work, namely aspects of the specific used finite element algorithm, a validation study, an interpolation scheme proposed for the specific use of large scale seismic scenarios given in terms of velocity models and a large scale simulation of the seismic response of the Aburra Valley using a recently developed crust model for the sedimentary basin underlying the city of Medellin and its main metropolitan area.

1.1 Literature Survey

There are three main numerical methods used for large scale simulations of earthquakes: (i) the finite difference method, (FDM); (ii) the finite element method, (FEM); and (iii) the spectral element method, (SEM). The boundary element method is also an important method for the simulation of the elastic wave equation but it has not been used by many researchers in large scale simulations because it's difficult to parallelize, it's limited for highly heterogeneous domains and it's impractical when considering material non-linearities. The literature survey will examine the three numerical methods often used in large scale simulations of earthquakes and their appropriateness to solve large scale wave propagation problems in earthquake engineering. Each method has had its main contributors who have championed its use and have been determinant to solve the challenges posed by large scale simulations. For each of the methods we will review its strengths and how they compare to other methods.

The Finite Differences Method is the most well known and simple numerical method to approximate partial differential equations. Its simplicity allows big computational models to be implemented and parallelized without much effort; however, its main drawback is that it needs a regular mesh as input. This is the conclusion of [1], that all approaches that account for heterogeneous models are difficult to implement and have to be adapted to the specific problem. FDM is the most active numerical method for large scale simulations of earthquakes, and has had contributions from several researchers like [2] and [3]. The most important researcher in large scale earthquake simulations using FDM is Prof. Yifeng Cui. In [4] the challenges involved in a large scale simulation that required 1.8 billion nodes are described, including modeling, preprocessing, solving and visualizing. In [5], a Magnitude 8 hypothetical earthquake in San Andreas' fault was simulated with the method, solving a model that required a uniform mesh of about 436 billion $40m^3$ cubes. The finite difference method has also seen advantage of using GPUs, computer processors targeted at graphics, to accelerate the simulations, obtaining a 77 fold increase in performance as shown in [6]. Even when the number of elements used might show FDM as a very powerful and scalable method, it can be seen that the number of degrees of freedom depend not only on the frequency and size of the simulation, but also on the materials involved. The lowest wave velocity for the model used in [5] was 400m/s; however, in the models usually studied, wave velocities go from 400m/s to 4500m/s. A model discretized using different wave velocities, instead of only the smallest, would require up to a thousand times less elements. Having several element sizes to discretize a region can give us the same results with less computational costs.

The Spectral Element Method is a modified version of the finite element method, (expanded in the next paragraph), where special collocation points, called Gauss-Lobatto-Legendre, are used. The biggest advantage of the method is that it's possible to have higher order elements than in regular FEM and that the mass matrix is trivially invertible, so it does not require lumping to be used with explicit methods. As a shortcoming, the method requires quadrilateral elements in the 2D case and hexahedral elements in the 3D case. Current hexahedral meshers are very limited to adapt to the differences in wave velocities that can occur in the earth, so the models used with SEM require a lot of symmetries, like a whole earth approximate model. The main researcher of large scale simulations using the spectral element method is Prof. Dimitri Komatitsch. In [7], he used SEM to simulate large earthquakes in a model of the entire earth that used 14.6 billion of degrees of freedom with 82 million spectral elements. The model solved the anelastic wave equation for solid regions, and the acoustic wave equation in terms of potential for liquid regions like the outer core, obtaining a high degree of conformance with their seismograms for the Alaska's earthquake of 2002 and Colombia's earthquake of 1997. For the earth model, it was shown that a huge number of elements was required in the center of the earth, not because the materials or non-linearities, but because of the mesh definition and symmetries. Some optimizations to the earth mesh have been created in [8] and [9] to have a mesh with less elements; however, they show that this process is not automatic and every mesh requires a human design around its symmetries. Other models have also been simulated with semi-regular meshes optimized for the SEM. One of this is Los Angeles basin, as shown in [10]. The simulation uses a mesh of 672,768 spectral elements distributed in 144 processors. The number of degrees of freedom is 136 million. The mesh generation considers that the surface requires more elements, but it still can't adapt to the differences of materials other than in the vertical direction. The method can also be easily optimized to work with GPUs, computer processors intended for graphics, as shown in [9]. The method is powerful and does not require lumping, or other approximations, but is as limited as the FDM by the requirements of the mesh.

The Finite Element Method is one of the most versatile numerical methods. It has the advantage that meshing is very easy as it can be formulated with almost any geometrical element and it can be easily coupled with other numerical methods. The explicit finite element method can be used to solve the dynamic elastic equation without assembling matrices, which can save memory and processing power. It also can be parallelized very easily when lumping the mass matrix. The most important researcher of large scale simulations of earthquakes using FEM is Prof. Jacobo Bielak. In [11] he worked in large scale simulations using tetrahedral meshes. Thetrahedral meshers are very fast, and can model any topography and ground features automatically; however, its disadvantage is that they require more elements than regular hexahedral meshes, (5 tedrahedra are required to create an hexahedron), and tetrahedra are less accurate to solve elastic problems than hexahedra, ([12]). Creating hexahedral meshes is much harder, so, as an alternative, FEM was modified to allow octree-based meshes in [13]. The octree starts with a uniform regular mesh composed of equal sized cubes. When the wave velocity at the location of the element requires it, the cube is divided into 8 equal sized elements. The process stops when all the elements represent the ground features completely. The octree uses a compatibility constraint in the nodes that are created in the transition from bigger to smaller elements, making different sized cubes the only kind of elements in the mesh. A complete description of the original method can be found in [14]. A slight variation of the algorithm to make it faster was made in [15], where matrix-vector multiplications are replaced with a method that requires less computations increasing the speed by 3 fold. The method, together with an scheme for creating the mesh, solving and extracting the visualization,

was used originally to simulate problems with 300 million degrees of freedom, which allowed the simulation of regions of 80 x 80 x 80 Km^3 and frequencies up to 1 Hz. The method has been shown to be scalable and has been used in many applications. In [16] it was used to propose an earthquake model from historical earthquake events. Also, in [17] the method solved a problem that required 5 billion elements with frequencies up to 4Hz. The method is the most versatile of all the numerical methods used to solve large scale simulations of earthquakes, having been modified to model the topography of a region in [18], something which can't be done with FDM or SEM. As a conclusion, the method has shown good results and its versatility makes it very appropriate for the generalization of large scale simulations to different problems in regions where the materials and the topography are determinant.

1.2 Appropriation of Large Scale Simulations for Wave Propagation Problems Technology

In its basic form, the modelling and simulation of wave propagation problems for earthquake engineering consists of the following four steps:

1. First a model of the problem with its geometry, properties and initial conditions is created.
2. The model is converted into a set of computer data structures which hold its discretized information. This is called pre-processing.
3. A solver of the numerical method is run to find an approximate solution to the problem.
4. The solution is presented to the user in a way in which he can do further calculations or take decisions. This is called post-processing.

Each of these four steps provides challenges in large scale simulations that will be addressed in this thesis. A diagram of the tasks that need to be addressed can be found in figure 1.1:

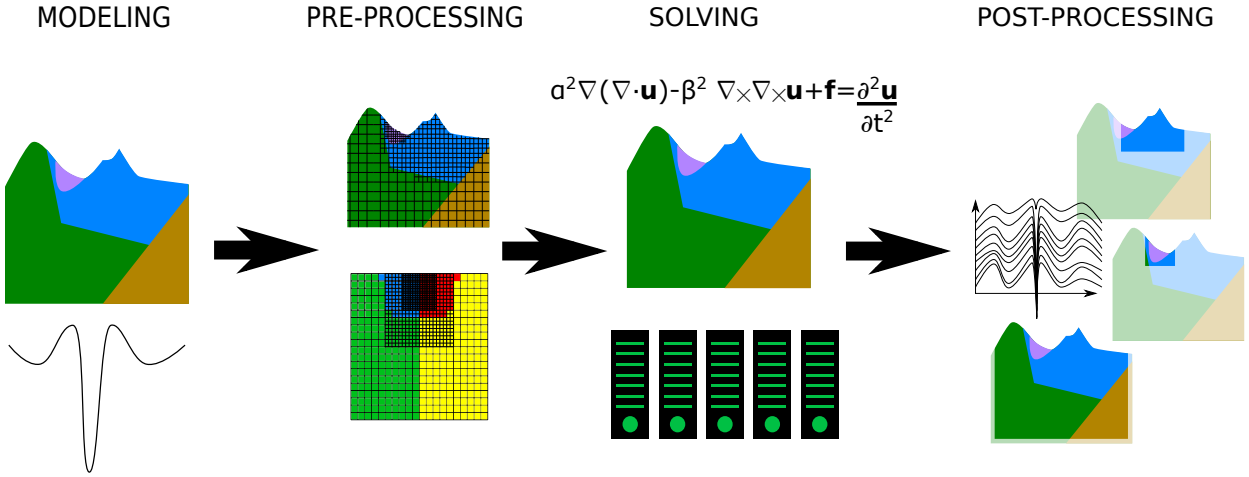


Figure 1.1: Tasks necessary to solve large scale simulations of wave propagation problems in earthquake engineering

DAMIAN-PAR is the name of the software created by EAFIT University for the large scale simulation of wave propagation problems in earthquake engineering. DAMIAN-PAR considers the pre-processing, solving and post-processing in a single program because all of the steps need a cluster to be done correctly. The model of Aburra's Valley, used to test the implementation was created as a separate program, a Community Velocity Model, which not only can be used with DAMIAN-PAR, but can be used with other simulation software. It was used with 3D simulation software in [18]. The following section presents a brief outline of the thesis.

The numerical method chosen in the current thesis was first introduced by Bielak's group, which in 2D is a quadtree instead of an octree in 2D. The reason is not only the close relationship between EAFIT and CMU, but also that Bielak's method is the most appropriate to model Aburra's Valley, which has an aggressive topography and material properties that change very fast. A topography would be very difficult to model with FDM or SEM. Instead, FEM works well with it. The changes in material properties would make FDM and SEM not optimal, FEM can adapt to heterogeneities easily and the quadtree makes it even easier.

The Community Velocity Model, (CVM), was based on an interpretative model of Aburra's Valley provided by the Geology Department at Universidad EAFIT and compiled in [29] in the form of slices. Then a new interpolation method was developed to have a very smooth transformation between them so the properties would be defined at every point of the geometry. The software developed, called Aburra's Valley CVM, is also described here.

1.3 Outline of the Master of Science Thesis

The Thesis contains 6 chapters as follows:

1. **Introduction:** It presents the reasons why the project for the appropriation of large scale simulations of wave propagation problems in earthquake engineering was carried, a literature survey of numerical methods available to make large scale simulations of earthquakes, and an introduction to the components developed.

2. **Theoretical Framework and Strategy of DAMIAN-PAR:** It presents the different components that constitute DAMIAN-PAR and with the theoretical background behind the tools and the strategies chosen to be able to simulate large scale problems.

3. **Validation of DAMIAN-PAR with Classical Problems:** Four classical problems are used to validate DAMIAN-PAR and its approximations. First, low frequency problems are validated against the Boundary Element Method for accuracy. Then, high frequency problems are tested for scalability and performance.

4. **Interpolation of 2D Cuts of Earth Formations. Application to the Creation of Aburrá's Valley's Community Velocity Model:** This chapter describes the implementation of Aburra's Valley Community Velocity Model. It's ordered as an article and it's self contained. It describes the interpolation method which had to be developed in order to interpolate a set of slices into a full 3D model that describes the elastic properties and topography of Aburra's Valley.

5. **Results of Simulating Slices of Aburra's Valley:** A set of slices of Aburra's Valley CVM was simulated with DAMIAN-PAR and its results are shown. The scope of the thesis does not include an analysis of the results.

6. **Conclusions and Future Work:** The final chapter shows which knowledge was acquired during the appropriation and implementation of the CVM and DAMIAN-PAR. It also shows what might be the problems with the current methods and what future work needs to be taken after having appropriated the large scale simulation technology at EAFIT.

Chapter 2

Theoretical Framework and Strategy of DAMIAN-PAR

DAMIAN-PAR is the software developed in this thesis to appropriate Large Scale Simulations of Wave Propagation Problems in Earthquake Engineering. DAMIAN-PAR integrates the tasks in Fig. 2.1: (i) pre-processing, (ii) solving and (iii) post-processing. Pre-processing involves reading the configuration of the problem, generating the mesh and applying the boundary conditions and forces to the different nodes and elements. Solving means finding the values that approximate the solution of the partial differential equation at the nodes of the mesh, at the given times. Post-processing means extracting the solution in formats which can be visualized or analyzed further by a human.

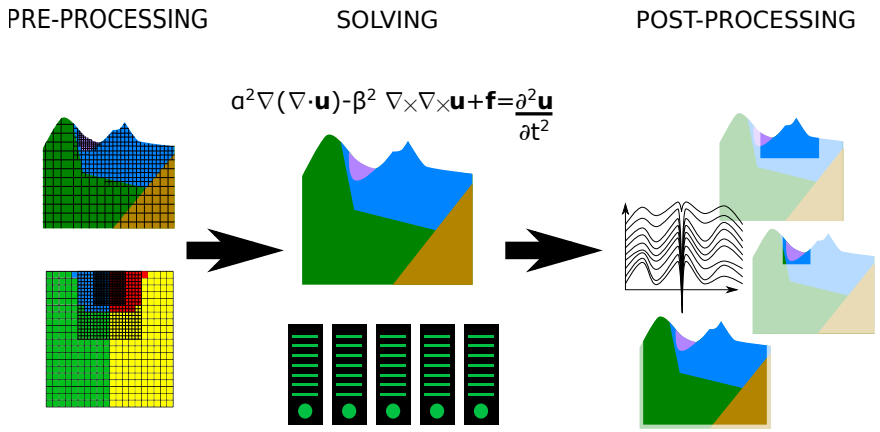


Figure 2.1: Tasks solved by DAMIAN-PAR

Large scale simulations can only be solved with the help of parallel computation, in which several computers run a subset of a simulation. As the size of the simulation increases, several processors are required to even hold the mesh in memory, so pre-processing, solving and post-processing need to be all parallel software. The components are integrated because clusters use queues to reserve the resources and execute the parallel programs that are sent to them in batch. All the data generated by the programs that is not saved appropriately is deleted from the servers

in which the task was executed.

Large Scale Simulations are required when frequencies are too high, feature sizes are too large or material properties are too complex or have low wave velocities. Here the following dimensionless frequency in terms of frequency f, shear wave propagation velocity β and characteristic dimension of the scatterer r is defined like:

$$f_o = \frac{fr}{\beta}. \tag{2.1}$$

The dimensionless frequency can describe the relative size of a problem with a single parameter. This parameter can be generalized to other problems. In general, the number of elements in the FEM method discretization is proportional to:

$$\left(\frac{f \cdot FS}{\beta}\right)^D \tag{2.2}$$

where FS is the size of the feature studied (e.g., a valley o canyon) and D is the dimension of the problem, in this case 2. In realistic seismic scenarios, very high dimensionless frequencies are possible when the size of the feature is very large or the materials have a very low wave velocity.

This chapter is divided as follows:

1. Section 2.1 formulates the elastic wave equation with its initial and boundary conditions.
2. Section 2.2 shows the algorithms used to discretize the domain in a quad-tree manner.
3. Section 2.3 shows the finite element discretization of the problem, how it applies to the quad-tree and how loads and boundary conditions are applied.
4. Section 2.4 shows the two forms of post-processing supported by DAMIAN-PAR, generation of synthetic seismograms over determined surfaces (i.e., referred herein as time sheets) and creation of animations; and how they are implemented.

2.1 Problem Statement

Navier's equations of linear elastodynamics are used to model seismic wave displacements on the earth. Let u_i represent the vector field of the three displacement components; let λ and μ be Lamé's parameters; let ρ be the density distribution; let ρf_i be a time depended body force, which may represent, for instance a seismic source; let $\varepsilon_{ij} = \frac{1}{2}\left(u_{i,j} + u_{j,i}\right)$ be the strain tensor and let $\tau_{ij} = \lambda\varepsilon_{kk}\delta_{ij} + 2\mu\varepsilon_{ij}$ be the stress tensor. Let Ω be an open bounded domain in $\mathbb{R}^3$ with free surface Γ_{FS}, truncation boundary Γ_{AB}, and outward unit normal to the boundary n_i. The initial-boundary value problem is then written as:

$$\rho\ddot{u}_i - \mu u_{i,jj} - (\lambda+\mu)\, u_{j,ji} = \rho f_i \qquad \text{in } \Omega \times \{0 \le t \le T\} \tag{2.3}$$

$$\sigma = t_i n_i = a\rho\alpha\dot{u}^N \qquad \text{on } \Gamma_{AB} \times \{0 \le t \le T\} \tag{2.4}$$

$$\tau = \|t_i - \sigma n_i\| = b\rho\beta\dot{u}^T \qquad \text{on } \Gamma_{AB} \times \{0 \le t \le T\} \tag{2.5}$$

$$t_i = \tau_{ij} n_j = 0 \qquad \text{on } \Gamma_{FS} \times \{0 \le t \le T\} \tag{2.6}$$

$$u_i = 0 \qquad \text{on } \Omega \times \{t = 0\} \tag{2.7}$$

$$\dot{u}_i = 0 \qquad \text{on } \Omega \times \{t = 0\} \tag{2.8}$$

The truncation boundary is defined with the boundary conditions proposed in [19]. In the problem definition σ is the normal stress, τ is the shear stress, $\dot{u}^N$ is the normal particle velocity and $\dot{u}^T$ is the tangential particle velocity while a and b are dimensionless parameters, selected as 1.0 in [19]. P-waves propagate with velocity $\alpha = \sqrt{(\lambda + 2\mu)/\rho}$, and S-waves with velocity $\beta = \sqrt{\mu/\rho}$.

Most realistic seismic problems are approximated by 3D models; however, 2D-based models can be useful to represent certain scenarios by plane strain idealizations and can also be used to interpret an develop conceptual understanding of complex 3D-derived results. This can help us determine the appropriateness of our approximations because the solutions to classical problems are well studied. In plane strain problems all field variables are independent of x_3 and the displacement in the x_3-direction vanishes identically. We set $u_3 \equiv 0$ and $\partial/\partial x_3 \equiv 0$ in (2.3) to obtain the plane strain equation. Hooke's law yields the following relations:

$$\tau_{\alpha\beta} = \lambda u_{\gamma,\gamma}\delta_{\alpha\beta} + \mu\left(u_{\alpha,\beta} + u_{\beta,\alpha}\right) \tag{2.9}$$

$$\tau_{33} = \lambda u_{\gamma,\gamma} \tag{2.10}$$

Where greek indices can assume the values 1 and 2 only. The initial-boundary value problem in 2D is defined as:

$$\rho\ddot{u}_\alpha - \mu u_{\alpha,\beta\beta} + (\lambda + \mu)\, u_{\beta,\beta\alpha} = \rho f_\alpha \qquad \text{in } \Omega \times \{0 \le t \le T\} \tag{2.11}$$

$$t_\alpha = \tau_{\alpha\beta} n_\beta = 0 \qquad \text{on } \Gamma_{FS} \times \{0 \le t \le T\} \tag{2.12}$$

$$\sigma = t_\alpha n_\alpha = a\rho\alpha\dot{u}^N \qquad \text{on } \Gamma_{AB} \times \{0 \le t \le T\} \tag{2.13}$$

$$\tau = \|t_\alpha - \sigma n_\alpha\| = b\rho\beta\dot{u}^T \qquad \text{on } \Gamma_{AB} \times \{0 \le t \le T\} \tag{2.14}$$

$$u_\alpha = 0 \qquad \text{on } \Omega \times \{t = 0\} \tag{2.15}$$

$$\dot{u}_\alpha = 0 \qquad \text{on } \Omega \times \{t = 0\} \tag{2.16}$$

2.2 Pre-Processing

The preprocessing usually contains the meshing and application of the loads and boundary conditions. In Large Scale Simulations, it also contains the partition of the mesh into several processors that will run each one a part of the simulation, sharing the results among them.

A quadtree is a geometric tree data structure that divides the space partitioning each square cell recursively into four equal square cells. As a geometric data structure is a very powerful tool to divide the space and execute geometric queries very fast. As a mesh data structure, the quadtree has been used as a basis for finite element approximation in [20]. Elements are divided into four equal elements until local refinement criteria are satisfied. In Fig. 2.2, the discretization scheme is illustrated.

In the case of seismic wave propagation in heterogeneous media, the criterion for meshing is that there are at least p nodes per local wavelength. As the shear wave velocity is smaller than the pressure wave velocity, the size of the element should be $h_e < \frac{1}{p}\frac{\beta}{f_{max}}$. Algorithm 1 creates the quad-tree using a model of the wave velocity given a point.

When the grid is refined, some nodes can end up being located in the middle of an element at a coarse to fine grid interface. Those nodes are called hanging nodes and the nodes around a hanging node are called anchor nodes. In Fig. 2.2, hanging and anchor nodes can be seen. An additional condition to create the mesh is that a hanging node can't be at the same time an anchor node, or said in another way, the element size can't differ more than two times between neighbor elements.

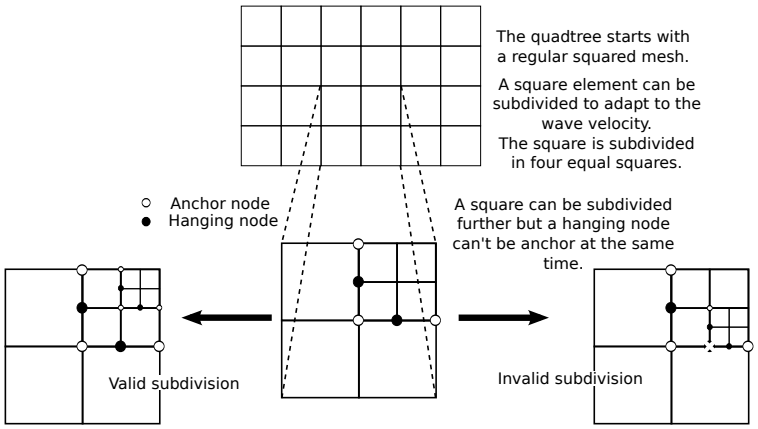


Figure 2.2: Mesh discretized with a Quadtree scheme

Data: Box: the box defined by coordinates: x_1, x_2, y_1, y_2, limiting the region to be meshed;
$VM\,(p)$: The velocity model, returns β at the point or air if is above the topography;
f: the maximum frequency of the pulse that will be simulated;
$N_{per\lambda}$: the number of elements per wave length
Result: M: the quad-tree mesh
begin initialize the mesh
  Let β_{max} be the highest S-wave velocity of the model;
  Let $Initial\_Size \leftarrow \frac{\beta}{f}\frac{1}{N_{per\lambda}}$;
  Function $Target\_Size\,(p) = \frac{VM(p)}{f}\frac{1}{N_{per\lambda}}$;
  Let $M \leftarrow \{S : S \text{ is a square with side } Initial\_Size \wedge S \in Box\}$;
end
repeat
  begin step that creates the quad-tree mesh
    for $S \mid \forall p \in S : VM(p)$ *is air* **do**
      $M \leftarrow M - \{S\}$;
    end
    for $S \mid p \in S \wedge Target\_Size(p) \leq$ *the side of the square* S **do**
      $M \leftarrow M - \{S\}$;
      $M \leftarrow M \cup \{$ the four equal squares dividing $S\}$;
    end
  end
until $\nexists S \in M \mid \forall p \in S \wedge Target\_Size(p) \leq$ *the side of the square* S;

Algorithm 1: Creates a quadtree mesh from a given velocity model

Algorithm 1 would not work for billions of nodes, as the memory required is not available in a single server, but in several of them. The elements have to be partitioned from the beginning and a subset has to be assigned to each processor in the cluster. The quadtree may require several subdivisions of the elements to get to the target element size, creating a node with several elements while others don't have so many, so the partition is repeated at every meshing step, so a single processor can hold its own subset in memory. Algorithm 2 creates the mesh in parallel.

Data: Box, x_1, x_2, y_1, y_2: the box limiting the region to be meshed;
$VM\,(p)$: returns β at the point or air if is above the topography;
f: the maximum frequency of the pulse that will be simulated;
$N_{per\lambda}$: the number of elements per wave length;
$P_1, P_2, P_3, ..., P_N$: the set of processors, P_L will be the local processor ;
Result: M: the distributed quad-tree mesh
begin Parallel initialization
    Let Box_L be a rectangular subset of Box assigned to P_L;
    Let M_L be the mesh initialized as in 1 using Box_L instead of Box;
end
repeat
    begin Parallel quad-tree loop
        Create the quad-tree mesh from M_L as in 1;
        Function $OD\,(S \in M)$ returns the processor to which the element belongs in an optimal partition;
        for $P_C \in \{P_1, P_2, ...P_N\} \wedge P_C \neq P_L$ **do**
            Send to P_C, $\{S \mid S \in M_L \wedge OD\,(S) = P_C\}$;
            Let $M_L \leftarrow M - \{S \mid OD\,(S) = P_C\}$;
            Let $M_L \leftarrow M \cup \{S \mid$ the elements received from $P_C\}$;
        end
    end
until $\nexists S \in M \mid p \in S \wedge Target\_Size(p) \leq$ *the side of the square* S;

Algorithm 2: Parallelization of the quad-tree mesh creation

The partition of the elements is not arbitrary, as a sum operation that needs communication by two processors is more expensive in time than a simple sum operation within each processor. If we look at the mesh as a graph, where an edge exists if two elements share a node, the partition is good if the edges cut are minimum, (the nodes shared by elements in different processors are minimum), while the number of vertices, (elements), in each subset is as even as possible. Parallel graph partition libraries are already available and provide algorithms like the k-way partition, where k is the number of processors. ParMETIS, ([24]), is the parallel library used by DAMIAN-PAR. The algorithm begins with an initial distribution of the nodes and then finds a good partition of the elements. A manual partition might be done if elements have to be deleted because they are above the topography, making some processors empty. The reason is that ParMETIS does not support empty processors.

2.3 Finite Element Method Solution to the Problem

The finite element method is used to solve equation 2.3 with bilinear square elements meshed in a quadtree manner. Upon spatial discretization, we obtain a system of ordinary differential equations of the form:

$$\boldsymbol{M}\ddot{\boldsymbol{u}} + \boldsymbol{C}^{AB}\dot{\boldsymbol{u}} + \boldsymbol{K}\boldsymbol{u} = \boldsymbol{f} \tag{2.17}$$

$$\ddot{\boldsymbol{u}}(\boldsymbol{0}) = \dot{\boldsymbol{u}}(\boldsymbol{0}) = \boldsymbol{0} \tag{2.18}$$

Where $\boldsymbol{M}$ and $\boldsymbol{K}$ are mass and stiffness matrices respectively, $\boldsymbol{f}$ is a body force vector resulting from a discretization of the seismic source model; and damping matrix $\boldsymbol{C}^{AB}$ represents the absorbing boundary as in [19]. We use a standard central difference scheme to represent displacement time derivatives yielding an explicit time-marching finite element algorithm. On the other hand, artificially diagonalizing the mass (and damping) matrices to decouple the global system of discrete finite element equations allows us to obtain the following FEM equations:

$$\begin{aligned}\left[\boldsymbol{M}_{diag} + \frac{\Delta t}{2}\boldsymbol{C}^{AB}_{diag}\right]\boldsymbol{u}^{t+\Delta t} =& \boldsymbol{f} - \left[\boldsymbol{K} - 2\frac{1}{\Delta t^2}\boldsymbol{M}\right]\boldsymbol{u}^{t} \\ &- \left[\frac{1}{\Delta t^2}\boldsymbol{M} - \frac{1}{2\Delta t}\boldsymbol{C}^{AB}\right]\boldsymbol{u}^{t-\Delta t}\end{aligned} \tag{2.19}$$

to be solved for $\boldsymbol{u}^{t+\Delta t}$ using information at t and $t-\Delta t$. In the above algorithm Δt is chosen such that the P-waves, which are the fastest, traverse an element in n steps. Then $\Delta t < \frac{1}{n}\frac{h_e}{\alpha}$.

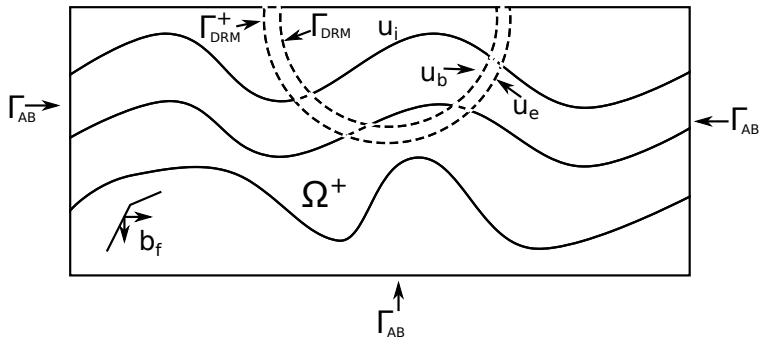
The discontinuity generated by the hanging nodes is resolved by constraining their field to be the mean of their anchor nodes. This creates a residual that should be added to the anchor nodes. In the case of the explicit finite element method, the following steps have to be added:

1. Sum half of the reactions of the hanging nodes to each neighbor anchor node after the right hand side has been calculated.

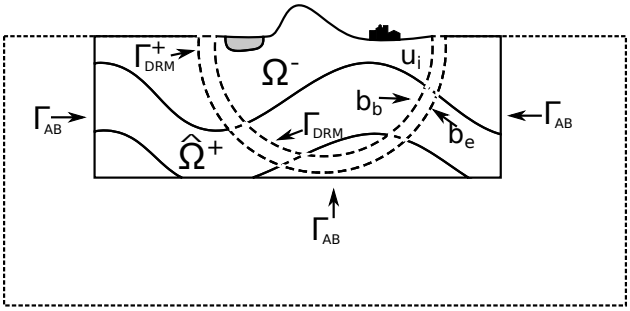
2. Set the displacements of each hanging node as the average of its anchor nodes after $\boldsymbol{u}^{t+\Delta t}$ has been calculated.

The term $\boldsymbol{f}$ was used together with a domain reduction method proposed in [21] to incorporate incoming P and SV plane waves at different angles. Plane waves have been extensively studied to understand the scattering in terms of basic wave phenomena, and benchmarked solutions are readily available for many plane wave problems. The solutions to plane wave problems hitting different features, like canyons or valleys, can be used to analyze the appropriateness of our approximations. The DRM is summarized in Fig. 2.3. Most of the following equations have been taken from [21], but some have been modified after our analysis and experiments.

The DRM is a two step method. In the first step, the displacement field $\boldsymbol{u}$ is calculated for a domain Ω from an input force $\boldsymbol{f}_s$ that represents a seismic event, and saved in a small frame of elements near Γ_{DRM}. Fig. 2.3a shows Γ_{DRM} which encloses a small region where displacements will be calculated. In the second step, the domain is truncated and the input forces are replaced by effective forces near Γ_{DRM}, called $\boldsymbol{f}_b$ and $\boldsymbol{f}_e$. The domain Ω can be truncated further such with this method, and the inside region can be modeled with a numerical method that allows nonlinearities.



(a) Domain Reduction Method step 1 scheme



(b) Domain Reduction Method step 2 scheme

Figure 2.3: Domain Reduction Method

Fig. 2.3b shows a truncated domain where more complex features have surfaced that would require more complex numerical methods also. The equations that allow the calculation of $\boldsymbol{f}_b$ and $\boldsymbol{f}_e$ are the following:

$$\boldsymbol{f}_{eff} = \begin{bmatrix} \boldsymbol{f}_i \\ \boldsymbol{f}_b \\ \boldsymbol{f}_e \end{bmatrix} = \begin{bmatrix} 0 \\ -\boldsymbol{M}_{be}^{\Omega^+} \ddot{\boldsymbol{u}}_e{}^0 - \boldsymbol{K}_{be}^{\Omega^+} \boldsymbol{u}_e^0 \\ \boldsymbol{M}_{eb}^{\Omega^+} \ddot{\boldsymbol{u}}_b{}^0 + \boldsymbol{K}_{eb}^{\Omega^+} \boldsymbol{u}_b^0 \end{bmatrix} \tag{2.20}$$

As $\boldsymbol{M}$ is a diagonal matrix, $\boldsymbol{M}_{mn}\ddot{\boldsymbol{u}}_n$ is reduced to 0 if $m \neq n$ leaving previous formula as simply:

$$\boldsymbol{f}_{eff} = \begin{bmatrix} \boldsymbol{f}_i \\ \boldsymbol{f}_b \\ \boldsymbol{f}_e \end{bmatrix} = \begin{bmatrix} 0 \\ -\boldsymbol{K}_{be}^{\Omega^+} \boldsymbol{u}_e^0 \\ \boldsymbol{K}_{eb}^{\Omega^+} \boldsymbol{u}_b^0 \end{bmatrix} \tag{2.21}$$

In the case of our work, the displacements $\boldsymbol{u}_b$ and $\boldsymbol{u}_e$, were obtained from analytical solutions for plane SV and P waves incident to a half space, convoluted with Ricker's Wavelet as formulated in [22]:

$$Ricker(t) = \left(1 - 2\pi^2 f_{peak}^2 t^2\right) e^{\left(-\pi^2 f_{peak}^2 t^2\right)} \tag{2.22}$$

Ricker's wavelet is controlled by a single parameter f_{peak}, the peak frequency.

The parallelization of the partial differential equation explicit FEM discretization is very easy because of the way equation 2.19 is organized. First we take the right hand side:

$$\boldsymbol{f} - \left[\boldsymbol{K} - 2\frac{1}{\Delta t^2}\boldsymbol{M}\right]\boldsymbol{u}^t - \left[\frac{1}{\Delta t^2}\boldsymbol{M} - \frac{1}{2\Delta t}\boldsymbol{C}^{AB}\right]\boldsymbol{u}^{t-\Delta t} \tag{2.23}$$

and note that every term is associated with elements, for example $\boldsymbol{M}$ is the sum of the elemental matrices $\boldsymbol{M}_e$. $\boldsymbol{f}$ is also associated with elements in a small frame, from equation 2.21. As such 2.23 can be rewritten like:

$$\begin{aligned}
&\boldsymbol{f}_1 - \left[\boldsymbol{K}_1 - 2\frac{1}{\Delta t^2}\boldsymbol{M}_1\right]\boldsymbol{u}_1^t - \left[\frac{1}{\Delta t^2}\boldsymbol{M}_1 - \frac{1}{2\Delta t}\boldsymbol{C}_1^{AB}\right]\boldsymbol{u}_1^{t-\Delta t}+\\
&\boldsymbol{f}_2 - \left[\boldsymbol{K}_2 - 2\frac{1}{\Delta t^2}\boldsymbol{M}_2\right]\boldsymbol{u}_2^t - \left[\frac{1}{\Delta t^2}\boldsymbol{M}_2 - \frac{1}{2\Delta t}\boldsymbol{C}_2^{AB}\right]\boldsymbol{u}_2^{t-\Delta t}+\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad ...
\end{aligned} \tag{2.24}$$

Where the subscript represents the element number to which the matrix or vector is associated. This means that the set of elements can be subsetted into several processors that can make the sum independently, and then the full sum can be made afterwards. In the pre-processing step, elements are already assigned and balanced in each processor and the solution of the PDE can be started. The following are the steps considering the quadtree and hanging nodes:

1. Calculate the right hand side at each processor using equation 2.23 and the DRM.
2. Add reactions from other processors at shared nodes.
3. Divide hanging nodes reactions by 2 and add them to its anchor nodes, (local and shared between processors).
4. Calculate $\boldsymbol{u}^{t+\Delta t}$ using equation (2.19).
5. Set the displacements at the hanging nodes as the mean of the displacements at the anchor nodes, (local and shared between processors).

2.4 Post-Processing

The solution, like the mesh and the internal data created by the simulation, is just too big to be handled by the analyst's personal computer. The solution has to be transformed into something that can be easily handled by the analyst in formats which are of use for him. There are two modes in which the solution can be presented to the analyst in DAMIAN-PAR:

1. Animation mode: It gives the analyst a frame of the solution at a given precision. The frame can be of the whole region modeled or just of a small fraction. For example, the analyst might want to obtain an animation in the metropolitan area of Aburra's Valley and not in the whole 2D slice modeled. The solution is exported to ParaView where advanced visualizations can be created for further analysis.
2. Synthetic Seismograms, (sheets), mode: It gives the analyst the solution at the surface of the region modeled. The solution can be also of a part of the region, like in the animation mode. It's exported to Matlab where a few datasets can serve the analyst to transform the data for further analysis.

Both modes make use of the quad-tree as a geometrical tree data structure, (opposed to as a mesh data structure), which can make queries very fast. In the first mode, points in a rectangular grid are searched in several processors. We define the quadtree geometrical data structure as follows: Let QT be a data structure with 5 operations:

1. $center(QT)$: returns a point c which is the center of QT.
2. $upper\_right(QT)$: returns the quadtree that contains the elements at the upper right of $center(QT)$ or empty.
3. $upper\_left(QT)$: returns the quadtree that contains the elements at the upper left of $center(QT)$ or empty.
4. $lower\_right(QT)$: returns the quadtree that contains the elements at the lower right of $center(QT)$ or empty.
5. $lower\_left(QT)$: returns the quadtree that contains the elements at the upper left of $center(QT)$ or empty.

$center(QT)$ has to be chosen in a way that makes every one of its four quadtree branches disjoint, an element can't be divided by a coordinate of the center. A point can be searched in a quadtree at each processor with algorithm 3.

Data: p: Point to be searched;
QT: Quadtree organized in a datastructure;
Result: e_p: the element in which p is located
repeat
  Let $c \leftarrow center(QT)$;
  if *QT is a leaf element e* **then**
    if $p \in e$ **then**
      $e_p \leftarrow e$;
    end
    else
      $e_p \leftarrow \emptyset$
    end
    return
  end
  if $p(X) > c(X) \wedge p(Y) > c(Y)$ **then** Let $QT \leftarrow upper\_right(QT)$;
  ;
  if $p(X) < c(X) \wedge p(Y) > c(Y)$ **then** Let $QT \leftarrow upper\_left(QT)$;
  ;
  if $p(X) > c(X) \wedge p(Y) < c(Y)$ **then** Let $QT \leftarrow lower\_right(QT)$;
  ;
  if $p(X) < c(X) \wedge p(Y) < c(Y)$ **then** Let $QT \leftarrow lower\_left(QT)$;
  ;
until $QT = \emptyset$;
$e_p \leftarrow \emptyset$

Algorithm 3: Algorithm that searches a point in a Quadtree

The synthetic seismograms are the progression of the displacements at the surface of the model. As the mesh is arbitrarily partitioned, the points at the surface also have to be searched using the quadtree but with a modification to the algorithm. This time we are interested in finding the highest point of the quadtree. Algorithm 4 finds the highest element which contains the given X coordinate. The algorithm needs a stack which is a data structure with the following operations:

1. $pop(ST)$ returns the object at the top of ST and removes it from the stack.
2. $push(ST, A)$ inserts an object A at the top of the stack.
3. $empty(ST)$ returns true if the stack is empty.

Data: XC: Coordinate to be searched at the surface;
QT: Quadtree organized in a datastructure;
Result: e_x: the highest element that bounds X
begin
  Stack definition
end
Let QT be a stack of pointers to quadtrees.;
$push(ST, QT)$;
repeat
  Let $QT \leftarrow pop(ST)$;
  Let $c \leftarrow center(QT)$;
  if *QT is a leaf element e* **then**
    if $p \in e$ **then**
      $e_p \leftarrow e$;
      return
    end
  end
  if $XC > c(X)$ **then**
    $push(ST, lower\_right(QT)$;
    $push(ST, upper\_right(QT)$;
  end
  if $XC < c(X)$ **then**
    $push(ST, lower\_left(QT)$;
    $push(ST, upper\_left(QT)$;
  end
until $\neg Empty(ST)$;
$e_p \leftarrow \emptyset$

Algorithm 4: Algorithm that finds the highest point of a Quadtree given a coordinate

Having found a point in an element, this is transformed to local element coordinates and the value of the displacements at that point is found using the following formula:

$$\boldsymbol{u} = \boldsymbol{H}\hat{\boldsymbol{u}} \tag{2.25}$$

Where $\boldsymbol{H}$ is the linear interpolation matrix used to formulate the Finite Element Method and $\hat{\boldsymbol{u}}$ is the value of the displacements at the degrees of freedom, given a time.

Chapter 3

Validation of DAMIAN-PAR with Classical Problems

In this section we conduct low and high frequency validations of the numerical simulator. Validations at low frequencies are targeted at knowing how the correctness of the solution is affected by the different approximations made to run large scale simulations. Validations at high frequencies are aimed at showing how the program works when stressed with large problems, if it can scale to a large amount of processors and how the time of the solution is affected by using more processors.

3.1 Low Frequency Validations

The validation at low frequencies is done against DAMIAN-BEM-HS, a direct boundary element method software using rigorous half-space Green's functions developed by Grupo de Mecanica Aplicada at Unversidad EAFIT. BEM is often used in the literature as a benchmark for other methods when analytical solutions are not available because of its sem-analytic character imposed by the Green's functions and the fact that the radiation boundary condition is exactly satisfied without the need to introduce artificial truncation boundaries.The BEM code has been validated extensively inside Grupo de Mecanica Aplicada.

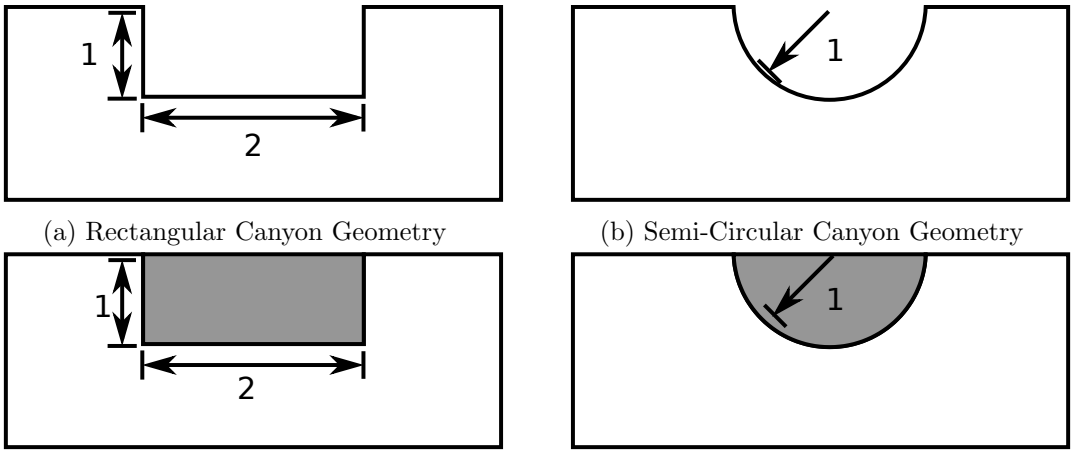
The four classic validations are shown in Fig. 3.1. The characteristics of the validations are:

1. The rectangular canyon: It's the most simple validation because it can be approximated perfectly using squared equal elements and because there's only one material to discretize which means the quad-tree and hanging nodes are not required.

2. The semi-circular canyon: It will show the magnitude of the error involved when discretizing a semi-circular free surface with perfect squares instead of following the path of the curve. It also has a single material to discretize as the rectangular canyon does. It does not use the quad-tree.

3. The rectangular valley: It can be approximated perfectly using squared elements but has two materials with different wave velocities which makes the quadtree necesary.

4. The semi-circular valley: It will show the error involved when discretizing a semicircular geometry with perfect squares instead of following the path of the curve and also will need the quadtree to model the interface between hard and soft materials.



(a) Rectangular Canyon Geometry

(b) Semi-Circular Canyon Geometry

(c) Rectangular Valley Geometry

(d) Semi-Circular Valley Geometry

Figure 3.1: Geometries for the validations

Vertical incidence S-Wave and P-Waves are used against all the models, giving 8 scenarios in total. The incident waves have velocity $\beta = 1.0$ for the hardest material and $\beta = 0.5$ for the softest one. Poisson's ratio is $v = 1/3$. The peak frequency of Ricker's pulse is $f_{peak} = 1.0$Hz while the maximum frequency is $f_{max} = 4.0$Hz. The amplitude of the incoming SV and P waves is 1.0. The program is dimensionless and consistency is left to the user.

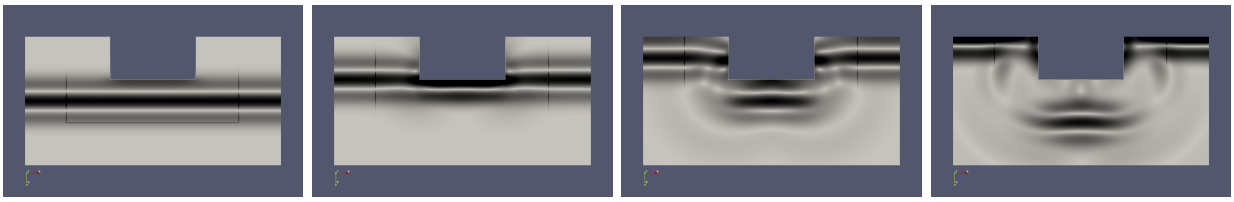
3.1.1 Rectangular Canyon

The simplest of the geometries to discretize with the quad-tree is the rectangular canyon. In this case there are four sources of diffraction located in each one of the corners of the canyon that will generate P and SV cylindrical waves and will also generate head waves connecting them. Grazing waves will be generated in the walls of the canyon.

Fig. 3.2 shows a few snapshots of the SV-wave hitting the rectangular canyon. In the graphics, the magnitude of the displacements is shown at four times while the wave is hitting the wall. Fig. 3.3 shows the synthetic seismograms of the displacements for several points in the surface of the canyon, making it look like a sheet. Fig. 3.4 shows the Fourier transform of the displacements for the rectangular canyon. The responses at the surface in both directions have been transformed using Fourier's transform and images have been generated for frequencies up to 4.0Hz, which is the maximum. Fig. 3.5 shows snapshots of the P-wave hitting the rectangular canyon, fig. 3.6 shows the synthetic seismograms of the displacements and fig. 3.7 shows the fourier transform. Comparisons made against BEM in the time domain and frequency domains are made after the previous figures. Fig. 3.8 shows the displacement in the time domain for 3 locations on the surface: X=0, X=1 and X=2 for the SV-Wave. Each response is centered at its X location. The solid line is the simulation with our software while the circles are the results with BEM. Fig. 3.9 shows the transfer function at frequency 1. The transfer function was obtained by transforming the response at the surface from the time domain to the frequency domain using Fourier's transform. After that, the response in the frequency domain which was nearer to 1 was divided by Ricker's pulse in the frequency domain at 1. The solid line shows our results while circles show BEM's result. Fig. 3.10 shows the comparison in the time domain for the P-wave, fig. 3.11 shows the comparison of the

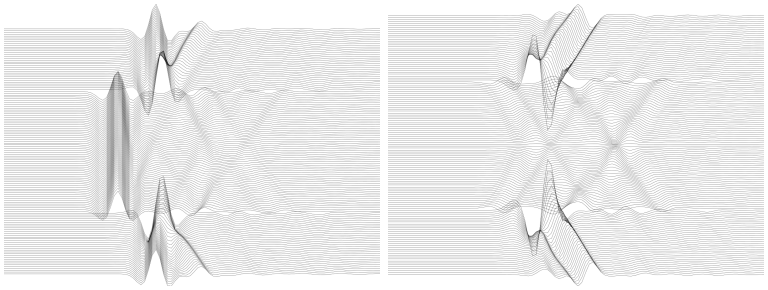
transfer function for the P-wave. In the following models similar methods will be used to obtain the images, however they will not be explained in detail.

The comparison in time domain shows a perfect correspondence between both DAMIAN-PAR and BEM. The transfer function shows also a very good correspondence with small differences that could be explained by the transformation of the time domain solution to frequency domain, because the frequency is not perfectly 1 but a frequency very close to it. In both cases it's shown that the rectangular canyon can be perfectly represented by the quadtree discretization, as expected.



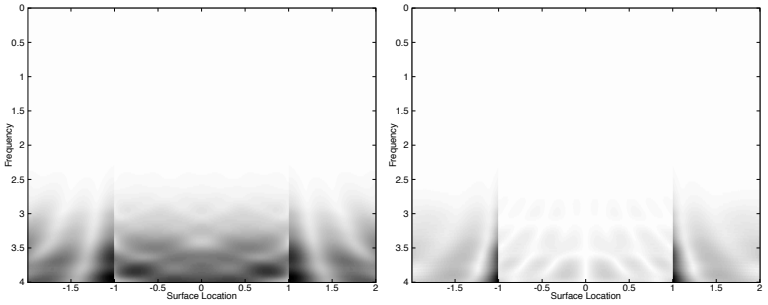
(a) $t = t_1$ (b) $t = t_2$ (c) $t = t_3$ (d) $t = t_4$

Figure 3.2: Snapshots of the SV-Wave incident to a rectangular canyon at 0 degrees.



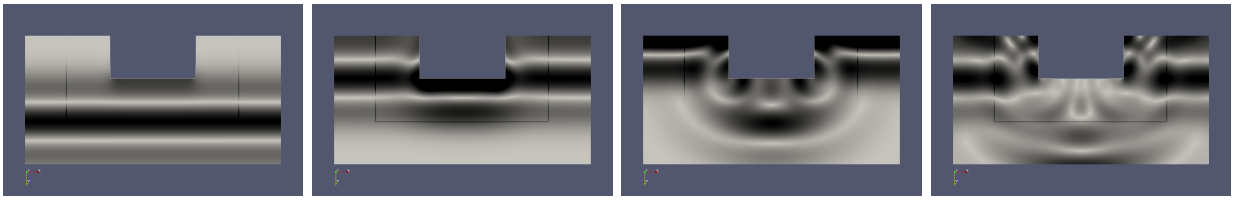
(a) Displacements in the X axis (b) Displacements in the Z axis

Figure 3.3: Synthetic seismograms of the SV-Wave incident to a rectangular canyon at 0 degrees.



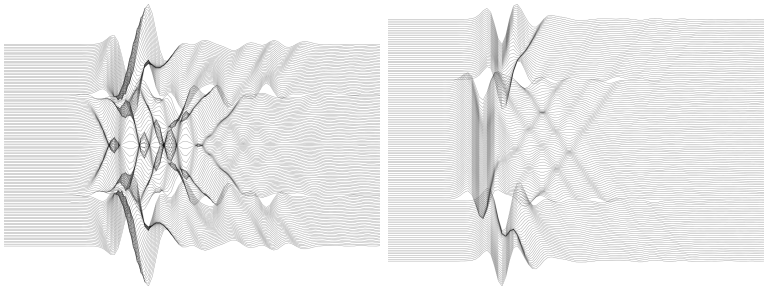
(a) Amplitude in the X axis (b) Amplitude in the Z axis

Figure 3.4: Fourier transform of the SV-Wave incident to a rectangular canyon at 0 degrees.



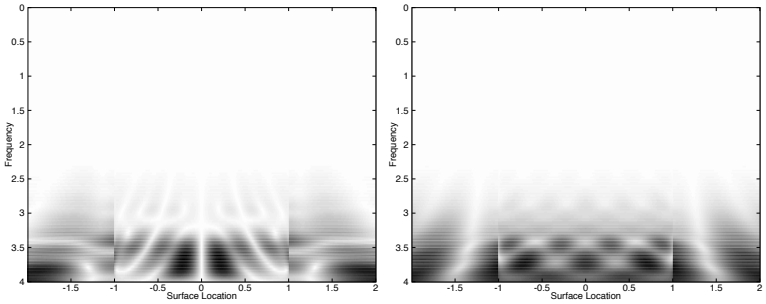
(a) $t = t_1$ (b) $t = t_2$ (c) $t = t_3$ (d) $t = t_4$

Figure 3.5: Snapshots of the P-Wave incident to a rectangular canyon at 0 degrees.



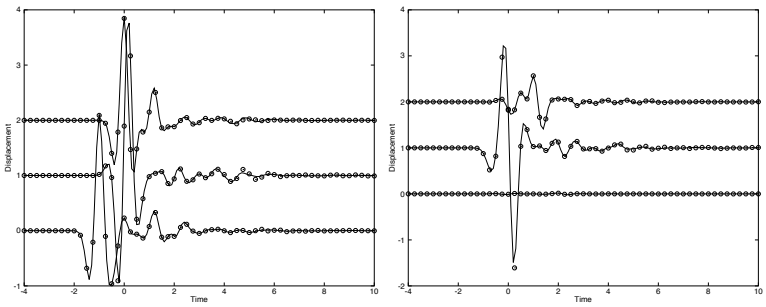
(a) Displacements in the X axis (b) Displacements in the Z axis

Figure 3.6: Synthetic seismograms of the P-Wave incident to a rectangular canyon at 0 degrees.



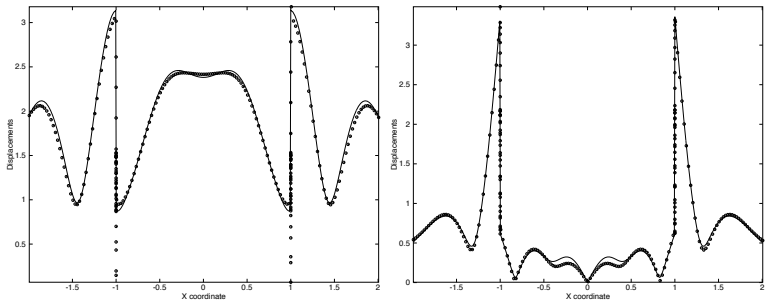
(a) Amplitude in the X axis (b) Amplitude in the Z axis

Figure 3.7: Fourier transform of the P-Wave incident to a rectangular canyon at 0 degrees.



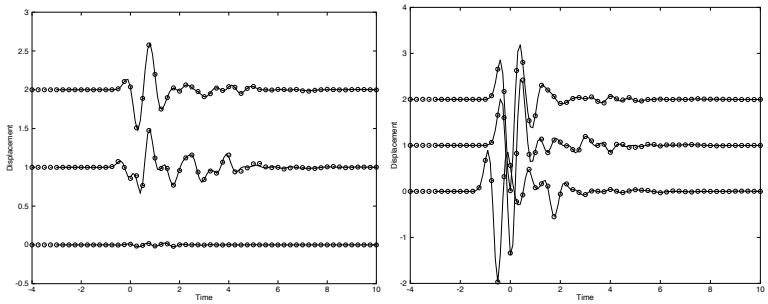
(a) Amplitude in the X axis (b) Amplitude in the Z axis

Figure 3.8: Time domain comparison of the SV-Wave incident to a rectangular canyon.



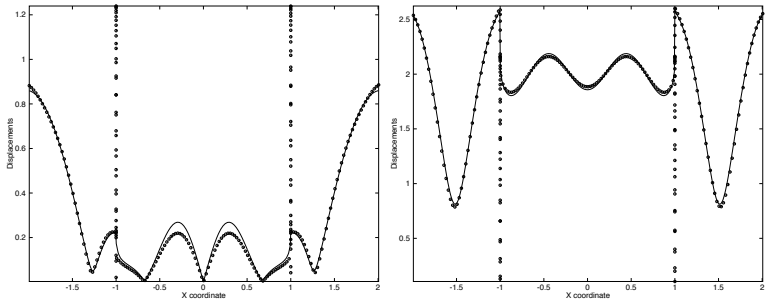
(a) Amplitude in the X axis (b) Amplitude in the Z axis

Figure 3.9: Transfer function comparison of the SV-Wave incident to a rectangular canyon.



(a) Amplitude in the X axis (b) Amplitude in the Z axis

Figure 3.10: Time domain comparison of the P-Wave incident to a rectangular canyon.



(a) Amplitude in the X axis (b) Amplitude in the Z axis

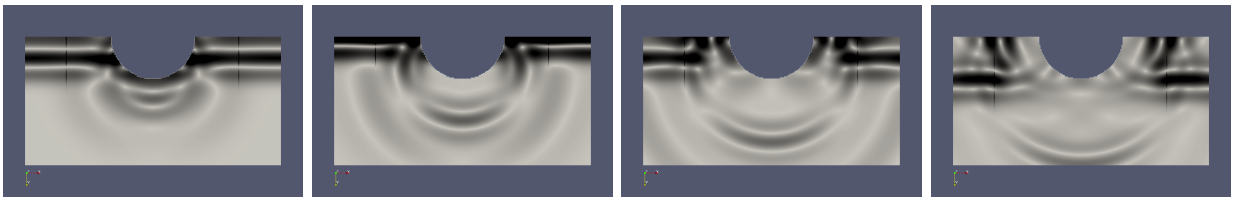
Figure 3.11: Transfer function comparison of the P-Wave incident to a rectangular canyon.

3.1.2 Semi-Circular Canyon

The geometry of the Semi-Circular Canyon is approximated with squared elements. In this case only two sources of diffraction located in each corner of the canyon exist. Almost cylindrical P and SV waves are generated by the canyon and they are connected by head waves.

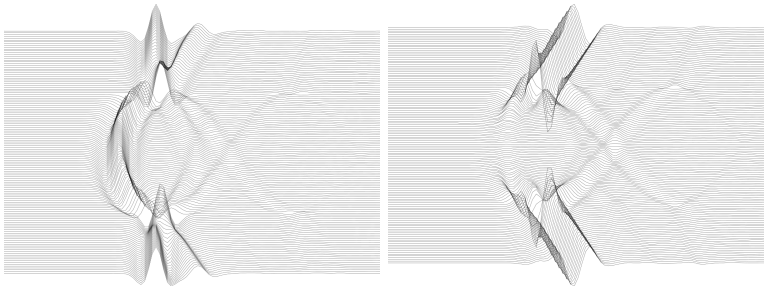
For the SV-wave hitting the semi-circular canyon, Fig. 3.12 shows a few snapshots, fig. 3.13 shows the synthetic seismograms and fig. 3.14 shows the Fourier transform. For the P-Wave, Fig. 3.15 shows a few snapshots, fig. 3.16 shows the synthetic seismograms and fig. 3.17 shows the Fourier transform. Fig. 3.18 shows the comparison for the SV-wave in the time domain while fig. 3.19 shows the comparison in the frequency domain. Fig. 3.20 shows the comparison for the P-wave in the time domain while fig. 3.21 shows the comparison in the frequency domain. The solid line is the simulation with explicit FEM while the circles are the results with BEM.

The comparison in time domain shows a good correspondence between both programs with smaller peaks for BEM. However, in frequency domain we can see that the transfer function of our program is not smooth, even when it's very close to the BEM function. It presents irregularities that might be explained by the squares approximation to the surface. This was expected for the canyon and it's the reason why Bielak's original software only worked for flat simulations. Aburra's Valley, which is being simulated, has a lot of topography. The solution looks bad at the canyon, but it does not look bad outside of it, and putting aside the lack of smoothness the solution, it's close enough to that of BEM.



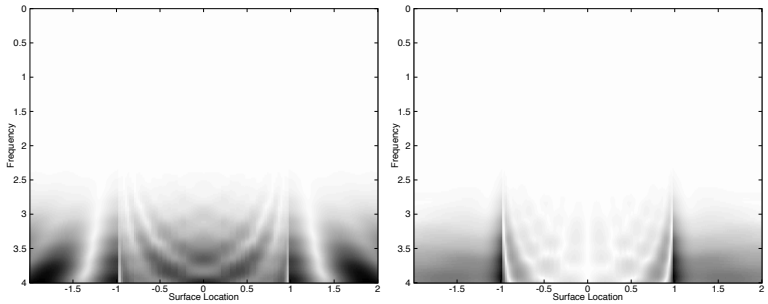
(a) $t = t_1$ (b) $t = t_2$ (c) $t = t_3$ (d) $t = t_4$

Figure 3.12: Snapshots of the SV-Wave incident to a semi-circular canyon at 0 degrees.



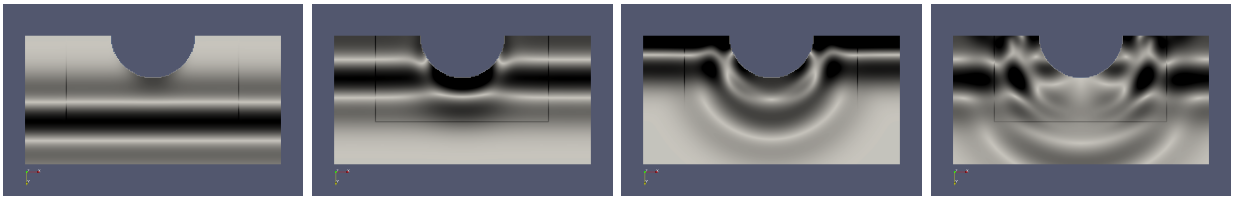
(a) Displacements in the X axis (b) Displacements in the Z axis

Figure 3.13: Synthetic seismograms of the SV-Wave incident to a semi-circular canyon at 0 degrees.



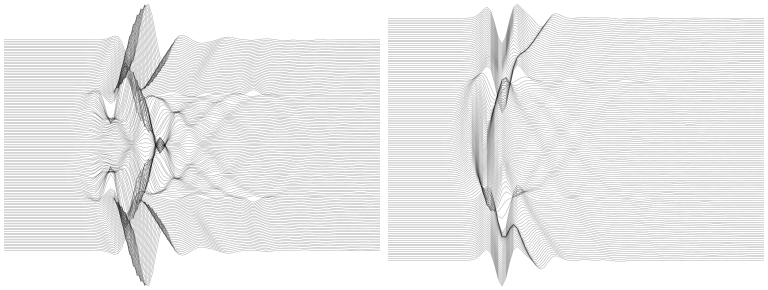
(a) Amplitude in the X axis (b) Amplitude in the Z axis

Figure 3.14: Fourier transform of the SV-Wave incident to a semi-circular canyon at 0 degrees.



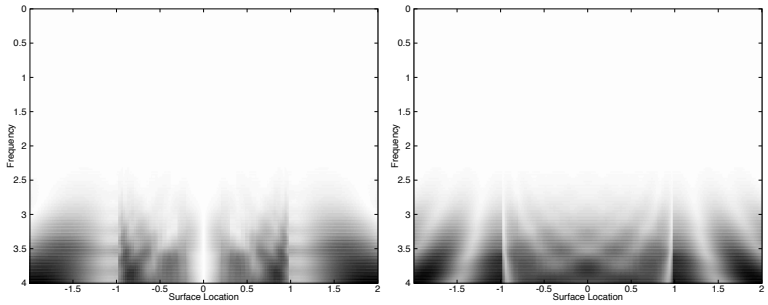
(a) $t = t_1$ (b) $t = t_2$ (c) $t = t_3$ (d) $t = t_4$

Figure 3.15: Snapshots of the P-Wave incident to a semi-circular canyon at 0 degrees.



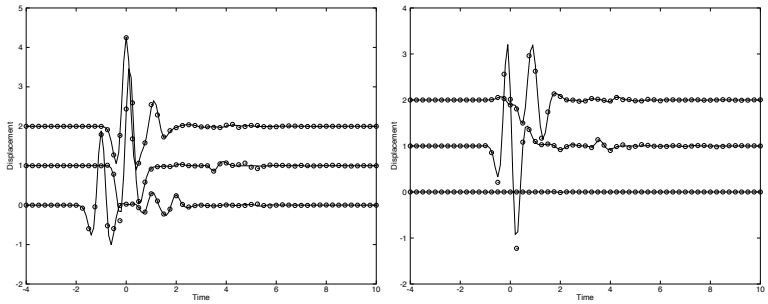
(a) Displacements in the X axis (b) Displacements in the Z axis

Figure 3.16: Synthetic seismograms of the P-Wave incident to a semi-circular canyon at 0 degrees.



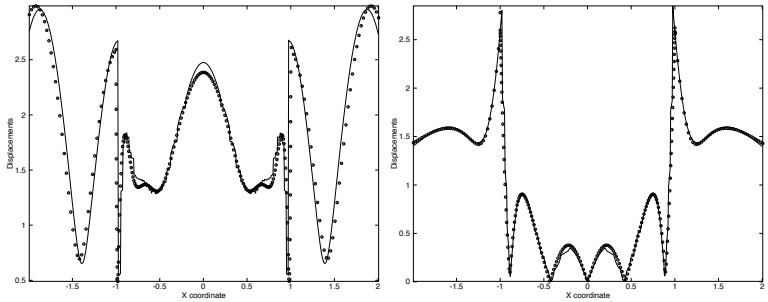
(a) Amplitude in the X axis (b) Amplitude in the Z axis

Figure 3.17: Fourier transform of the P-Wave incident to a semi-circular canyon at 0 degrees.



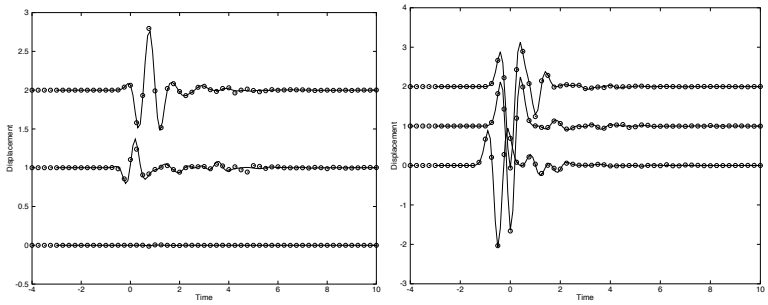
(a) Amplitude in the X axis (b) Amplitude in the Z axis

Figure 3.18: Time domain comparison of the SV-Wave incident to a semi-circular canyon.



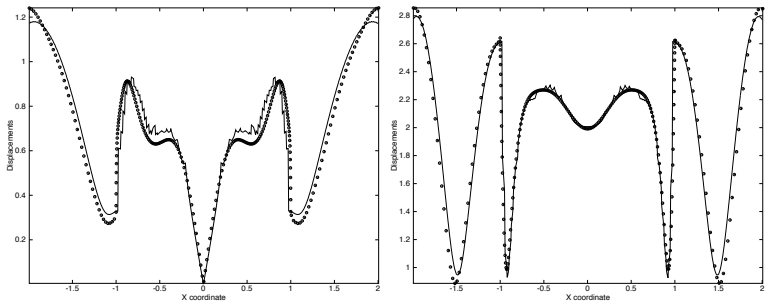
(a) Amplitude in the X axis (b) Amplitude in the Z axis

Figure 3.19: Transfer function comparison of the SV-Wave incident to a semi-circular canyon.



(a) Amplitude in the X axis (b) Amplitude in the Z axis

Figure 3.20: Time domain comparison of the P-Wave incident to a semi-circular canyon.



(a) Amplitude in the X axis (b) Amplitude in the Z axis

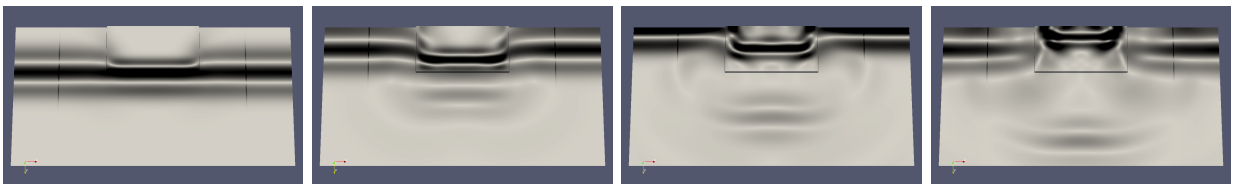
Figure 3.21: Transfer function comparison of the P-Wave incident to a semi-circular canyon.

3.1.3 Rectangular Valley

The geometry of the rectangular valley is matched perfectly by squares, but the interface between two materials is approximated using the quadtree scheme. In this case waves are reflected and refracted by the inner rectangle, which has a lower wave velocity. Four sources of diffraction are located in each of the corners of the rectangle and head waves connect the diffracted P and SV waves inside and outside the region. After the wave inside the lower velocity region is reflected by the free boundary, it hits again the lower border of the valley and gets reflected and refracted again. Waves generated by the sources of diffraction also remain in the valley being reflected and refracted so the wave requires a long time to be disappear completely.

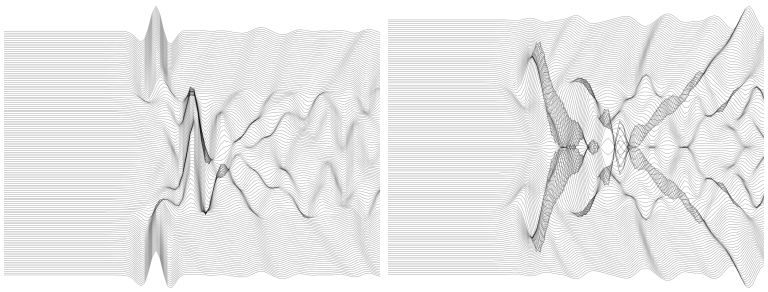
For the SV-wave hitting the rectangular valley, Fig. 3.22 shows a few snapshots, fig. 3.23 shows the synthetic seismograms and fig. 3.24 shows the Fourier transform. For the P-Wave, Fig. 3.25 shows a few snapshots, fig. 3.26 shows the synthetic seismograms and fig. 3.27 shows the Fourier transform. Fig. 3.28 shows the comparison for the SV-wave in the time domain while fig. 3.29 shows the comparison in the frequency domain. Fig. 3.30 shows the comparison for the P-wave in the time domain while fig. 3.31 shows the comparison in the frequency domain. The solid line is the simulation with explicit FEM while the circles are the results with BEM.

In time domain, the correspondence is good between our solution and BEM at the beginning, but starts to become less good with every reflection. This is magnified in the transfer functions, that show that the correspondence is not very good between BEM and DAMIAN-PAR. The discrepancies were not expected as the approximation to the geometry was supposed to be perfect and prove the correctness of the quadtree.



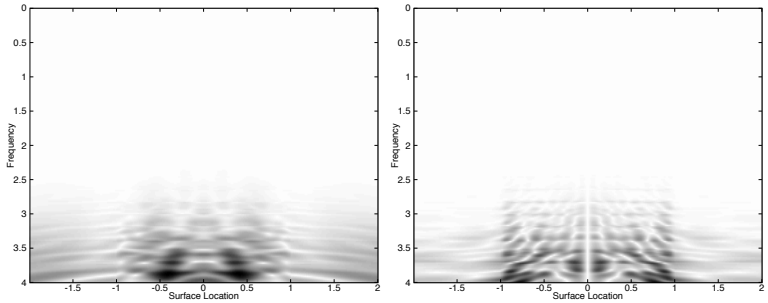
(a) $t = t_1$ (b) $t = t_2$ (c) $t = t_3$ (d) $t = t_4$

Figure 3.22: Snapshots of the SV-Wave incident to a rectangular valley at 0 degrees.



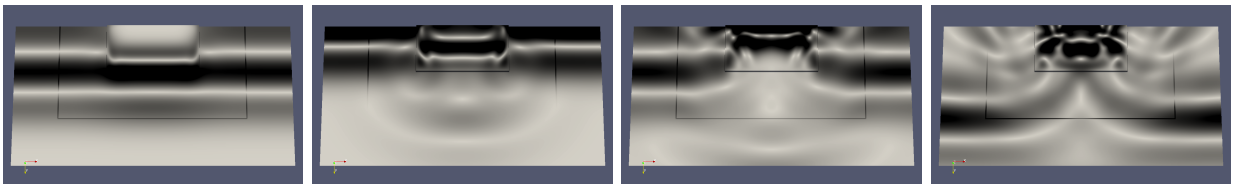
(a) Displacements in the X axis (b) Displacements in the Z axis

Figure 3.23: Synthetic seismograms of the SV-Wave incident to a rectangular valley at 0 degrees.



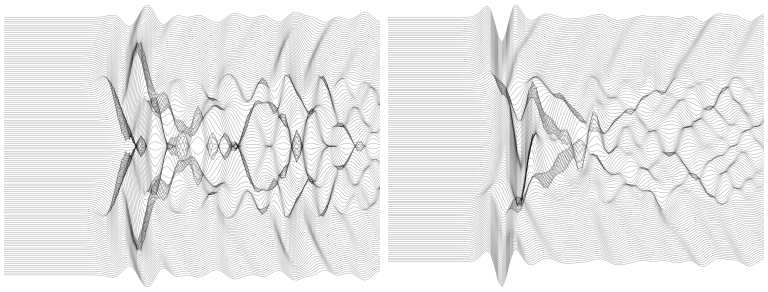
(a) Amplitude in the X axis (b) Amplitude in the Z axis

Figure 3.24: Fourier transform of the SV-Wave incident to a rectangular valley at 0 degrees.



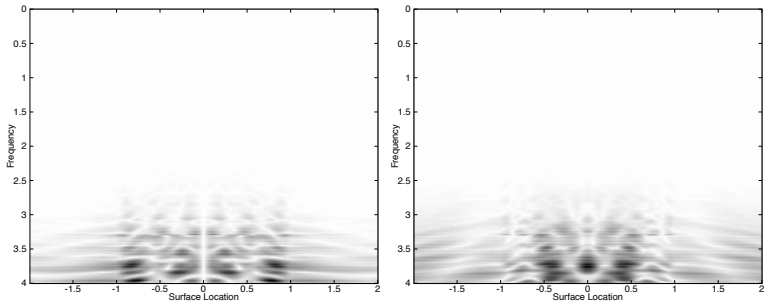
(a) $t = t_1$ (b) $t = t_2$ (c) $t = t_3$ (d) $t = t_4$

Figure 3.25: Snapshots of the P-Wave incident to a rectangular valley at 0 degrees.



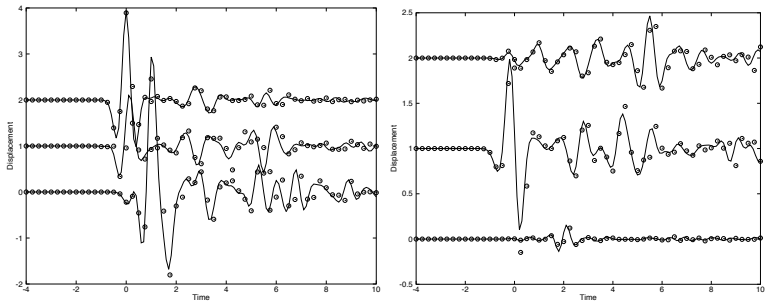
(a) Displacements in the X axis (b) Displacements in the Z axis

Figure 3.26: Synthetic seismograms of the P-Wave incident to a rectangular valley at 0 degrees.



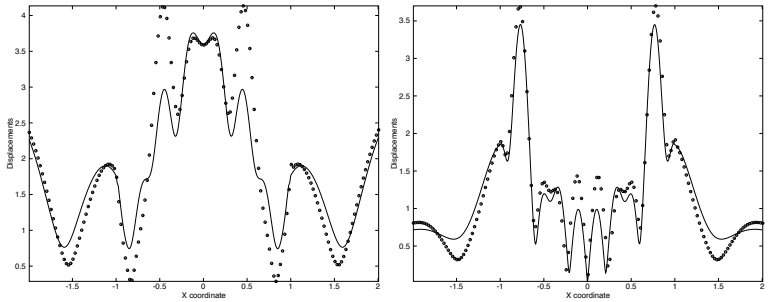
(a) Amplitude in the X axis (b) Amplitude in the Z axis

Figure 3.27: Fourier transform of the P-Wave incident to a rectangular valley at 0 degrees.



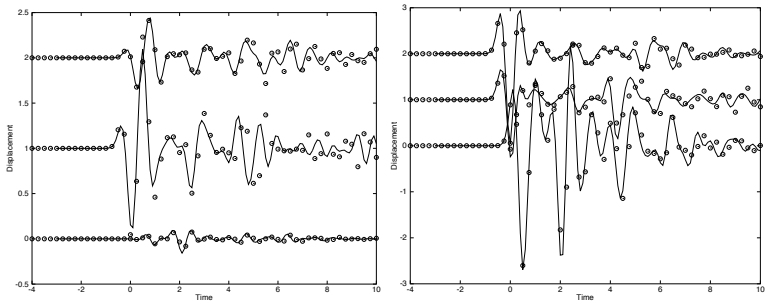
(a) Amplitude in the X axis (b) Amplitude in the Z axis

Figure 3.28: Time domain comparison of the SV-Wave incident to a rectangular valley.



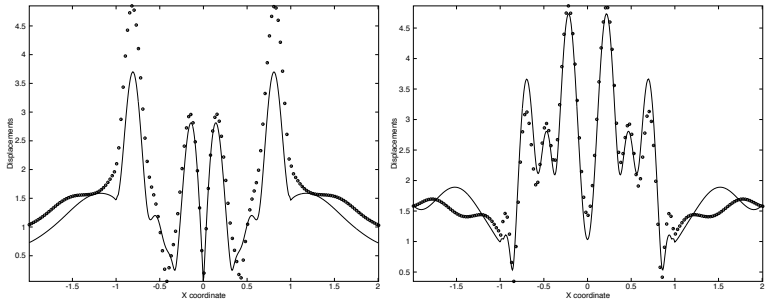
(a) Amplitude in the X axis (b) Amplitude in the Z axis

Figure 3.29: Transfer function comparison of the SV-Wave incident to a rectangular valley.



(a) Amplitude in the X axis (b) Amplitude in the Z axis

Figure 3.30: Time domain comparison of the P-Wave incident to a rectangular valley.



(a) Amplitude in the X axis (b) Amplitude in the Z axis

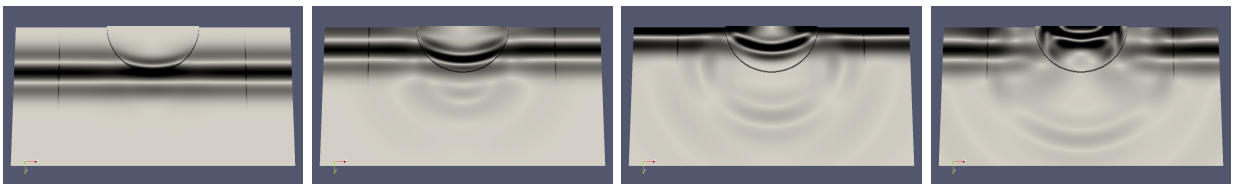
Figure 3.31: Transfer function comparison of the P-Wave incident to a rectangular valley.

3.1.4 Semi-Circular Valley

The geometry of the semi-circular valley is approximated with squares and the interface between both materials is approximated with the quadtree scheme. In this case waves are reflected and refracted by the inner semi-circle, which has a lower wave velocity. Refracted waves inside the lower velocity region create a concave wave. This model only has two sources of diffraction are located at each of the corners of the canyon. Like the rectangular canyon, the waves hit the boundary several times reflecting and refracting until they are weak enough.

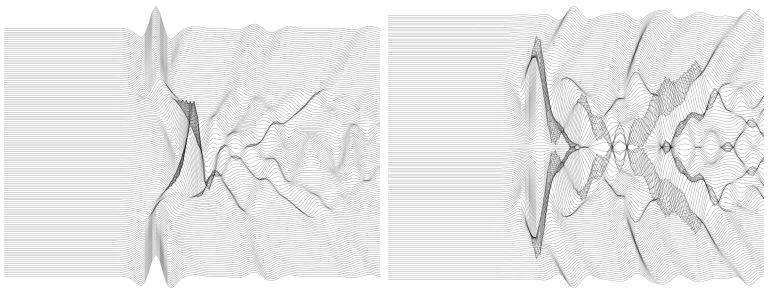
For the SV-wave hitting the rectangular valley, Fig. 3.32 shows a few snapshots, fig. 3.33 shows the synthetic seismograms and fig. 3.34 shows the Fourier transform. For the P-Wave, Fig. 3.35 shows a few snapshots , fig. 3.36 shows the synthetic seismograms and fig. 3.37 shows the Fourier transform. Fig. 3.38 shows the comparison for the SV-wave in the time domain while fig. 3.39 shows the comparison in the frequency domain. Fig. 3.40 shows the comparison for the P-wave in the time domain while fig. 3.41 shows the comparison in the frequency domain. The solid line is the simulation with explicit FEM while the circles are the results with BEM.

The correspondence is good between our solution and BEM at the beginning, but starts to become less good with every reflection. However, the solution does not seem to be worst than the solution of the rectangular valley, it's actually better as shown by the frequency domain comparison. This was unexpected because the semi-circular valley has the quadtree approximation and also models the interface as perfect squares. The approximation with squares is actually good enough to represent not squared geometries in this case, and not only because the interfaces are not well known as explained by Bielak.



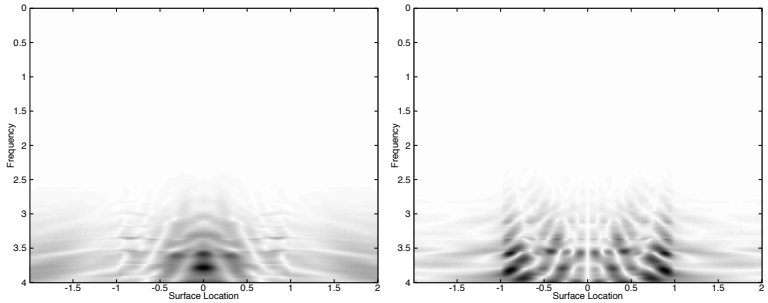
(a) $t = t_1$ (b) $t = t_2$ (c) $t = t_3$ (d) $t = t_4$

Figure 3.32: Snapshots of the SV-Wave incident to a semi-circular valley at 0 degrees.



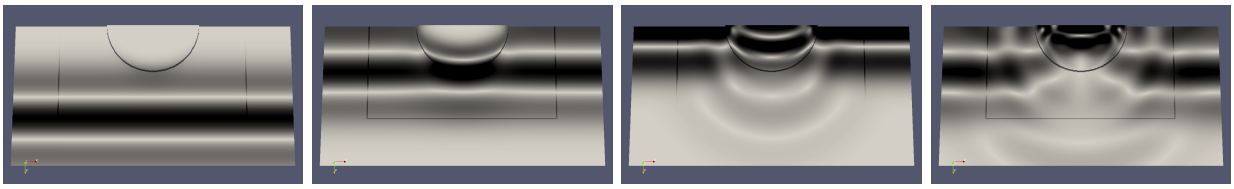
(a) Displacements in the X axis (b) Displacements in the Z axis

Figure 3.33: Synthetic seismograms of the SV-Wave incident to a semi-circular valley at 0 degrees.



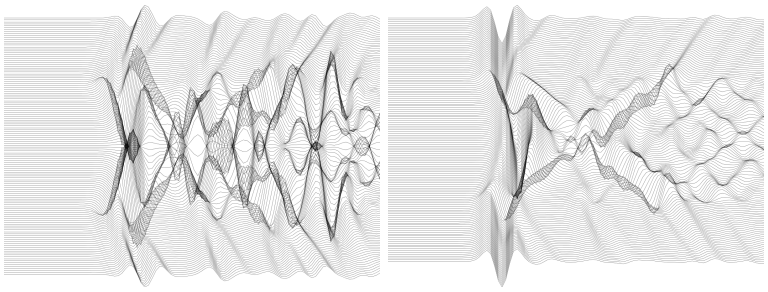
(a) Amplitude in the X axis (b) Amplitude in the Z axis

Figure 3.34: Fourier transform of the SV-Wave incident to a semi-circular valley at 0 degrees.



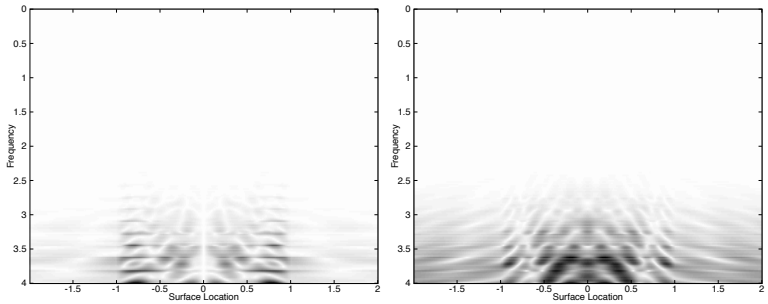
(a) $t = t_1$ (b) $t = t_2$ (c) $t = t_3$ (d) $t = t_4$

Figure 3.35: Snapshots of the P-Wave incident to a semi-circular valley at 0 degrees.



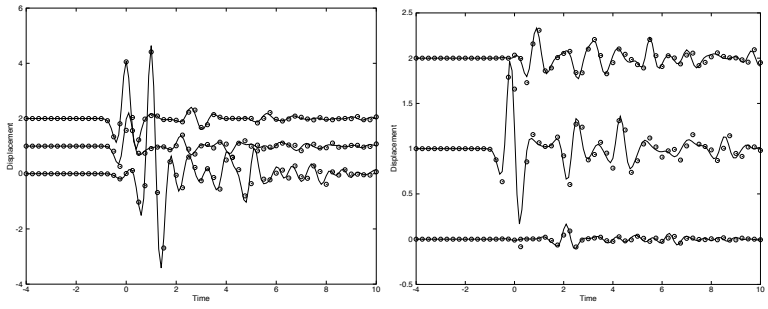
(a) Displacements in the X axis (b) Displacements in the Z axis

Figure 3.36: Synthetic seismograms of the P-Wave incident to a semi-circular valley at 0 degrees



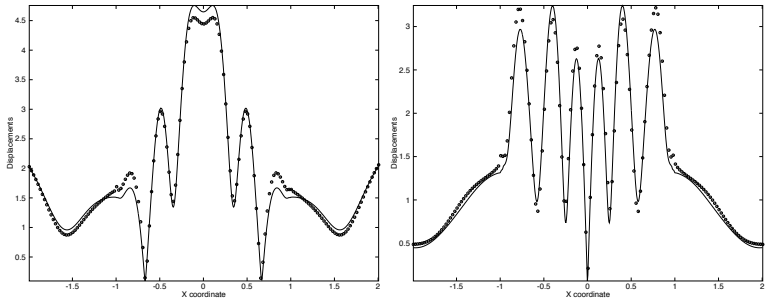
(a) Amplitude in the X axis (b) Amplitude in the Z axis

Figure 3.37: Fourier transform of the P-Wave incident to a semi-circular valley at 0 degrees.



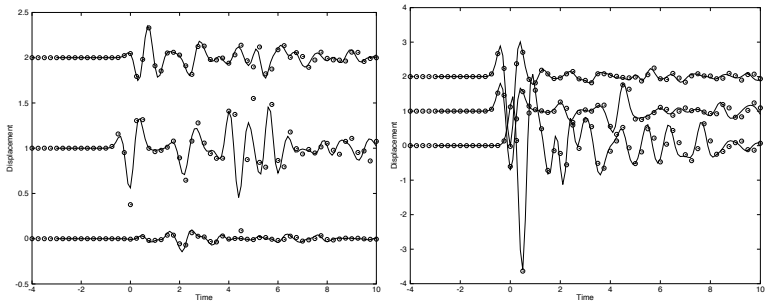
(a) Amplitude in the X axis (b) Amplitude in the Z axis

Figure 3.38: Time domain comparison of the SV-Wave incident to a semi-circular valley.



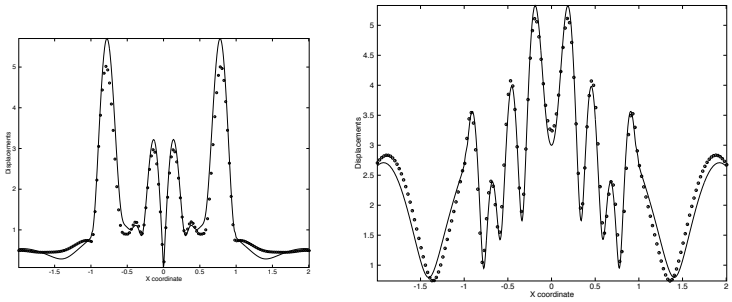
(a) Amplitude in the X axis (b) Amplitude in the Z axis

Figure 3.39: Transfer function comparison of the SV-Wave incident to a semi-circular valley.



(a) Amplitude in the X axis (b) Amplitude in the Z axis

Figure 3.40: Time domain comparison of the P-Wave incident to a semi-circular valley.



(a) Amplitude in the X axis (b) Amplitude in the Z axis

Figure 3.41: Transfer function comparison of the P-Wave incident to a semi-circular valley.

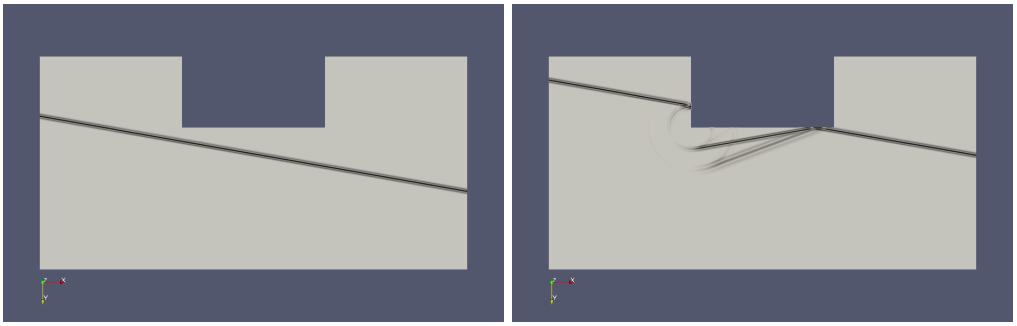
3.2 High Frequency Validations

At high frequencies what is important is the scalability of the solution. High frequencies are not realistic for most seismic events, as the range of frequencies usually goes up to 5 Hz. However, we are interested with the size of the problem as defined in section 2 which is related to the non-dimensional frequency. As such the dimensionless frequency for the semi-circular canyon is defined as:

$$f_0 = \frac{fr}{2\beta} \tag{3.1}$$

So given a problem it's possible that changed to a non-dimensional frequency it might seem unrealistic, but it's not when wave velocity is low or the feature's size is large. We made 8 simulations, this time with peak frequency $f_{peak} = 16$ and maximum frequency $f_{max} = 64$. We used 320 Quad-Core AMD Opteron®Processor 2380 to simulate the problem and set 4 hours of maximum time. We present the result only in snapshots, number of degrees of freedom, (DOFs), and time, as it's very difficult to analyze synthetic seismograms or other information at such high frequencies and BEM is not capable of simulating such high frequencies to compare.

Fig. 3.42 shows snapshots of the response of a rectangular canyon to an incident SV-Wave incident at 10 degrees. Fig. 3.43 shows snapshots of the response of a rectangular canyon to an incident P-Wave incident at 30 degrees. The progressions show more clearly the features of the rectangular canyon, which has sources of diffraction and shows the head waves generated by those sources more clearly. These were not easy to spot with lower frequencies. Fig. 3.44 shows snapshots of the response of a semi-circular canyon to an incident SV-Wave incident at 10 degrees. Fig. 3.45 shows snapshots of the response of a semi-circular canyon to an incident P-Wave incident at 30 degrees. It shows that the waves create an almost circular pattern like the described in the literature. Simulating a topography is the most important feature that the circular canyon helps to validate. Fig. 3.46 shows snapshots of the response of a rectangular valley to an incident SV-Wave incident at 10 degrees. Fig. 3.47 shows snapshots of the response of a rectangular valley to an incident P-Wave incident at 30 degrees. The response shows many reflections and refractions which might account for the discrepancies of the solution between BEM and our software as the time progresses shown by the low frequency validations. Fig. 3.46 shows snapshots of the response of a semi-circular valley to an incident SV-Wave incident at 10 degrees. Fig. 3.47 shows snapshots of the response of a semi-circular valley to an incident P-Wave incident at 30 degrees. The response shows how appropriate it is to use squares to represent non-squared features where the material changes because of the almost circular patterns formed.



(a) $t = t_1$

(b) $t = t_2$

(c) $t = t_3$

(d) $t = t_4$

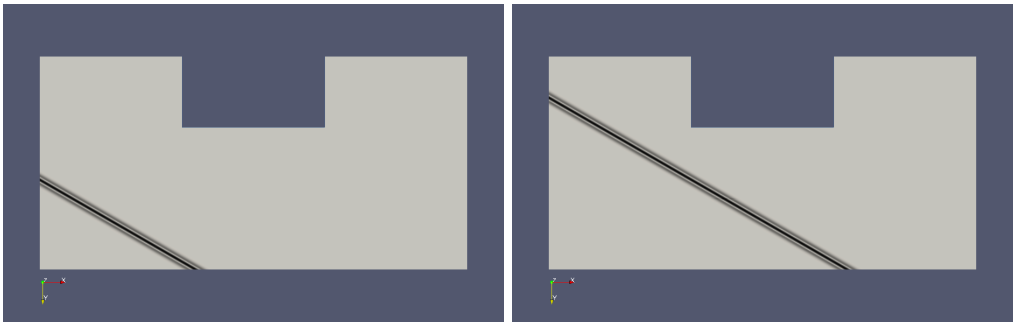
(e) $t = t_5$

(f) $t = t_6$

(g) $t = t_7$

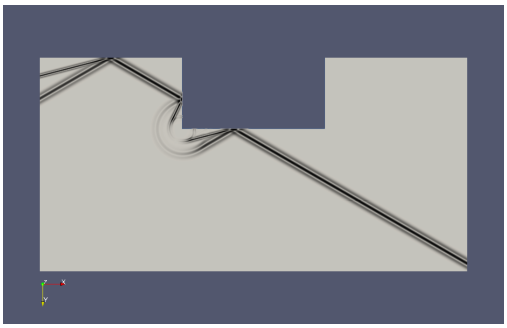
(h) $t = t_8$

Figure 3.42: Snapshots of the SV-Wave incident to a rectangular canyon at 10 degrees.

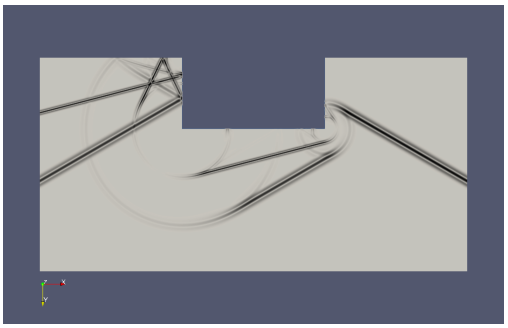


(a) $t = t_1$

(b) $t = t_2$



(c) $t = t_3$



(d) $t = t_4$

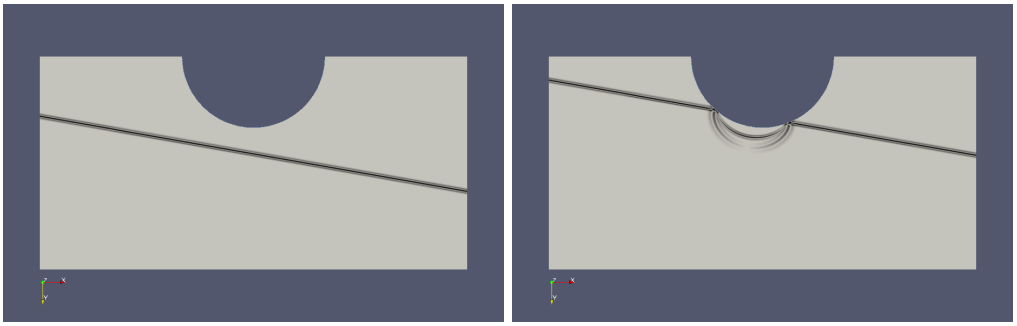
(e) $t = t_5$

(f) $t = t_6$

(g) $t = t_7$

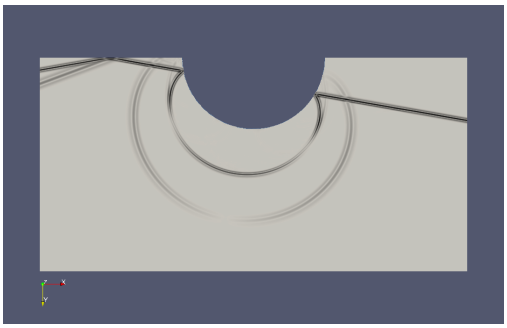
(h) $t = t_8$

Figure 3.43: Snapshots of the P-Wave incident to a rectangular canyon at 30 degrees.

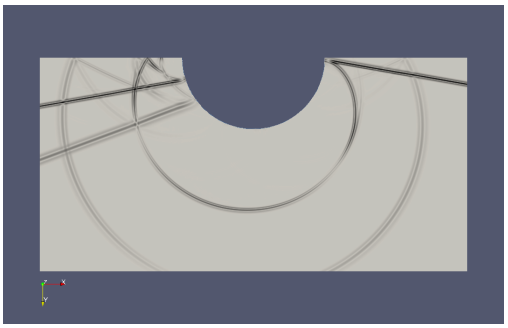


(a) $t = t_1$

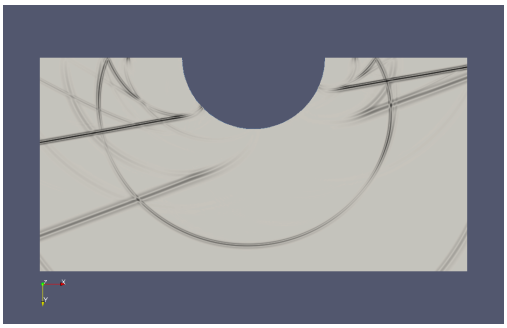
(b) $t = t_2$



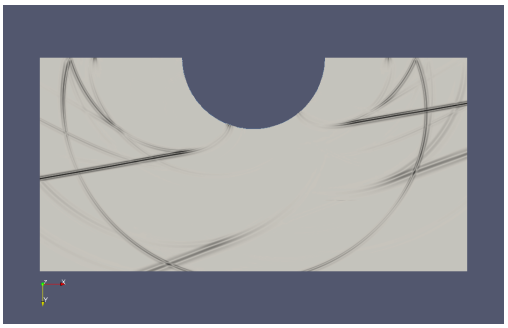
(c) $t = t_3$



(d) $t = t_4$



(e) $t = t_5$

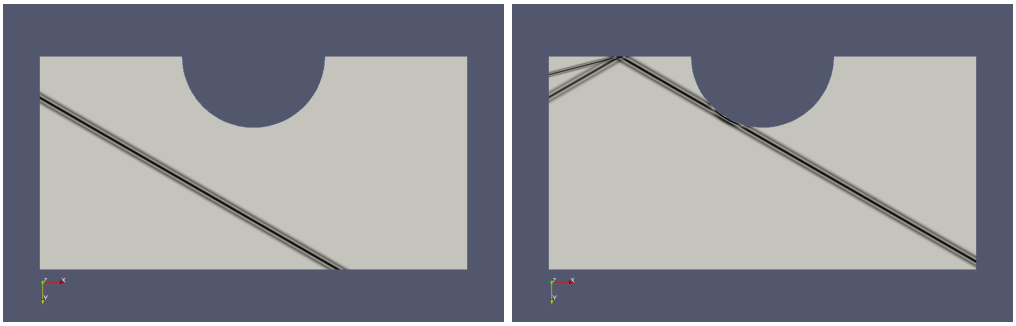


(f) $t = t_6$

(g) $t = t_7$

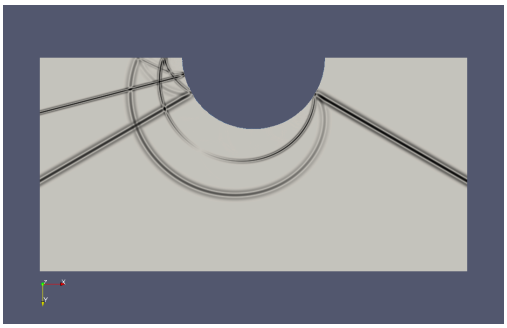
(h) $t = t_8$

Figure 3.44: Snapshots of the SV-Wave incident to a semi-circular canyon at 10 degrees.

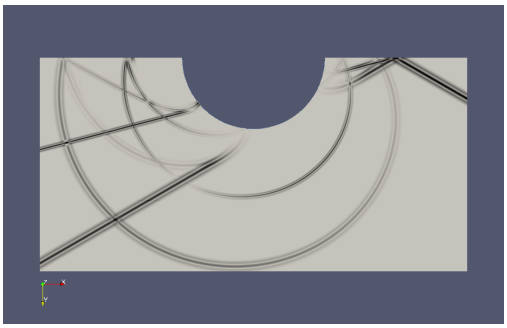


(a) $t = t_1$

(b) $t = t_2$

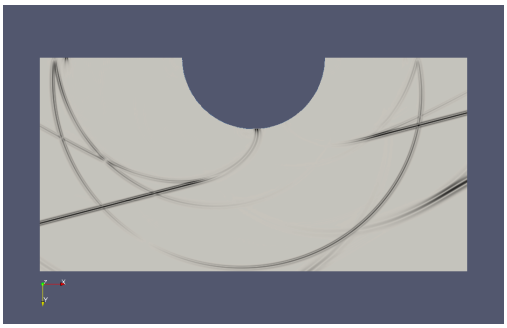


(c) $t = t_3$



(d) $t = t_4$

(e) $t = t_5$

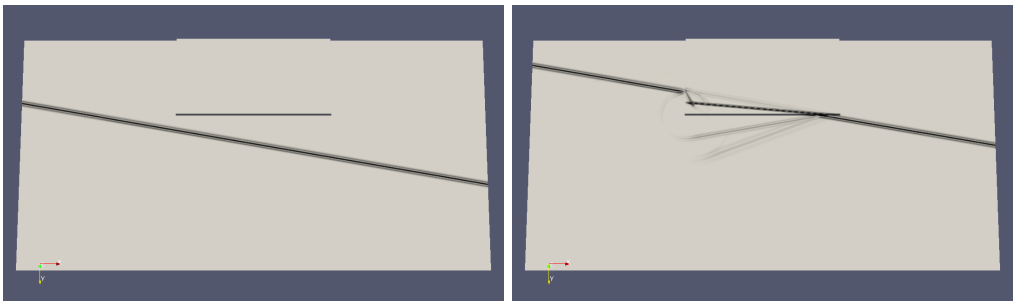


(f) $t = t_6$

(g) $t = t_7$

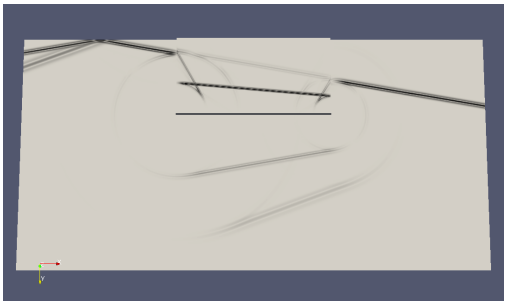
(h) $t = t_8$

Figure 3.45: Snapshots of the P-Wave incident to a semi-circular canyon at 30 degrees.

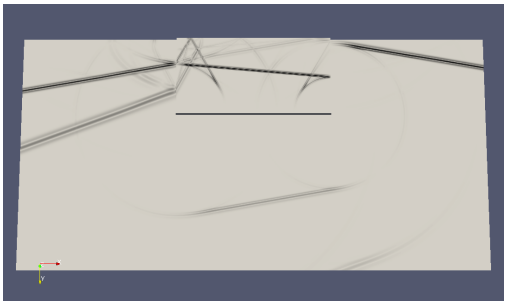


(a) $t = t_1$

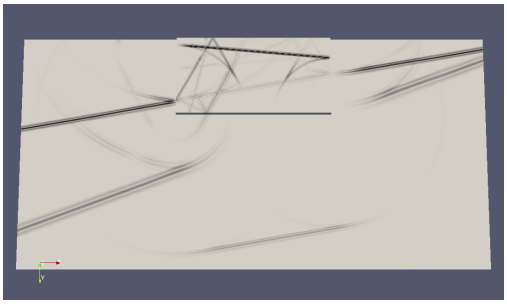
(b) $t = t_2$



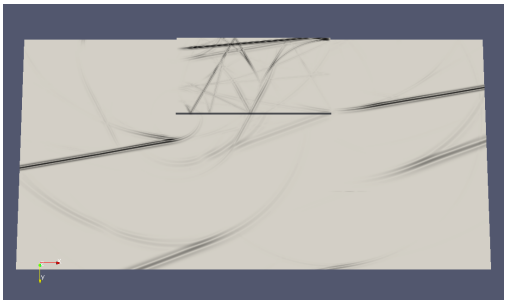
(c) $t = t_3$



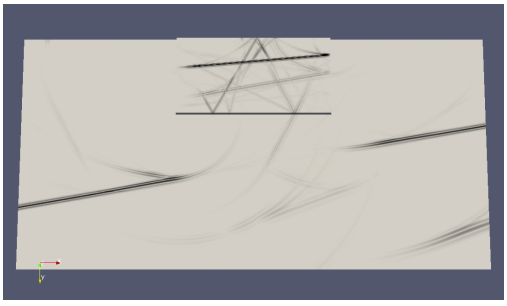
(d) $t = t_4$



(e) $t = t_5$



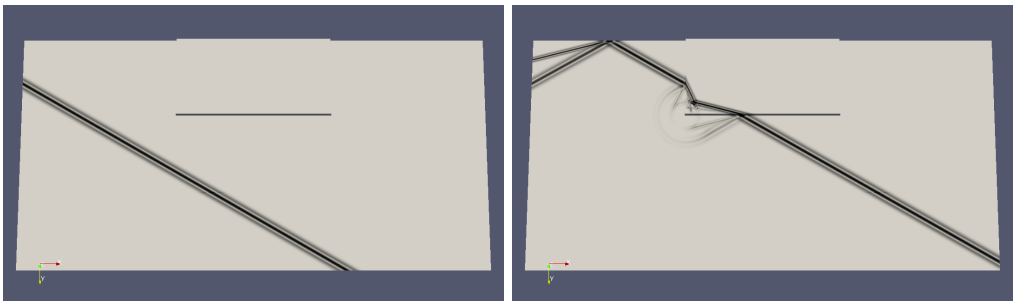
(f) $t = t_6$



(g) $t = t_7$

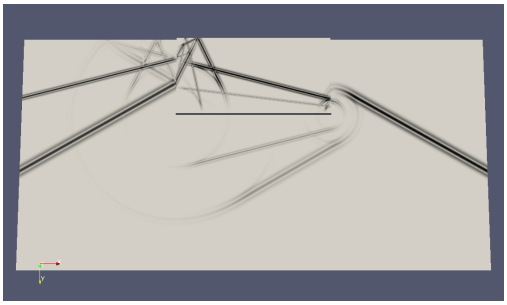
(h) $t = t_8$

Figure 3.46: Snapshots of the SV-Wave incident to a rectangular valley at 10 degrees.

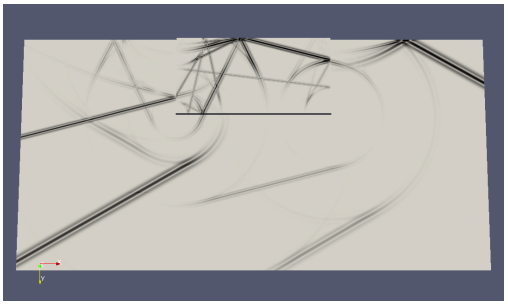


(a) $t = t_1$

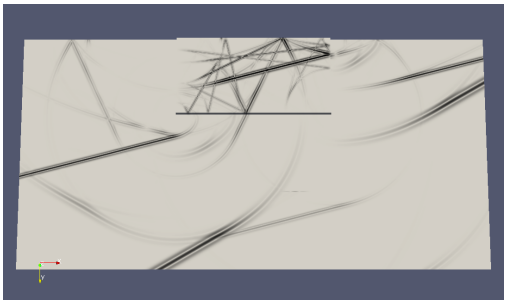
(b) $t = t_2$



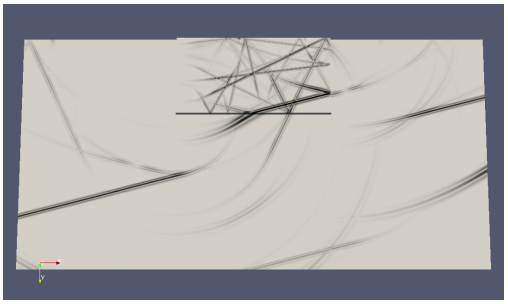
(c) $t = t_3$



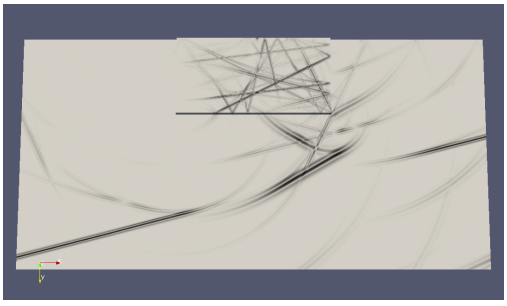
(d) $t = t_4$



(e) $t = t_5$



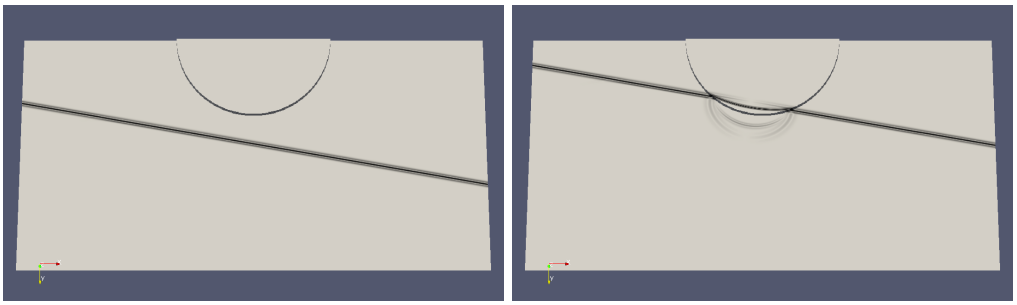
(f) $t = t_6$



(g) $t = t_7$

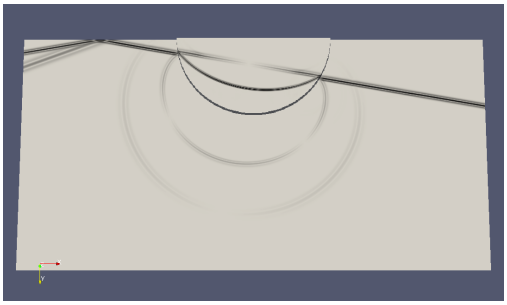
(h) $t = t_8$

Figure 3.47: Snapshots of the P-Wave incident to a rectangular valley at 30 degrees.

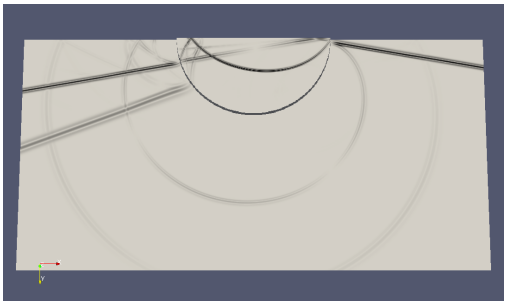


(a) $t = t_1$

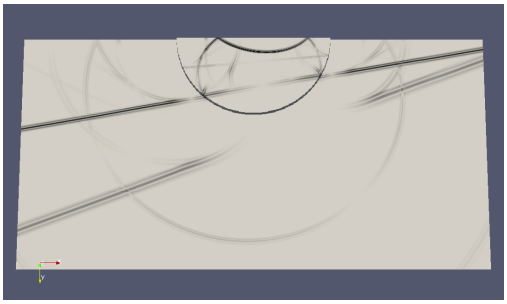
(b) $t = t_2$



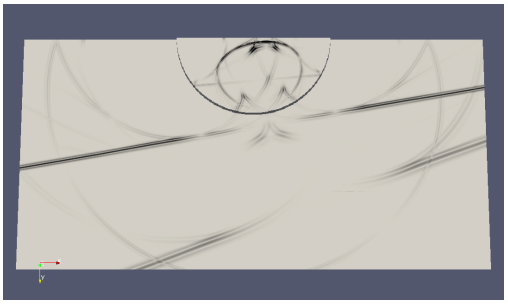
(c) $t = t_3$



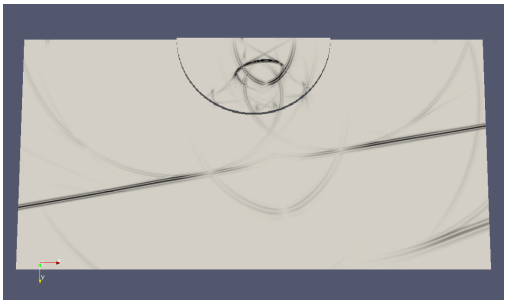
(d) $t = t_4$



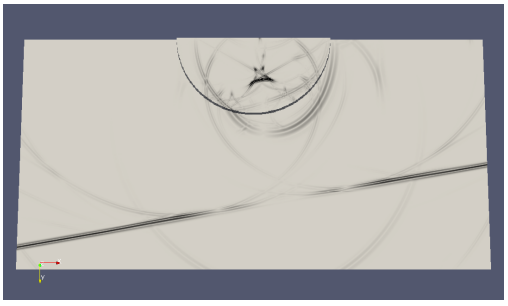
(e) $t = t_5$



(f) $t = t_6$

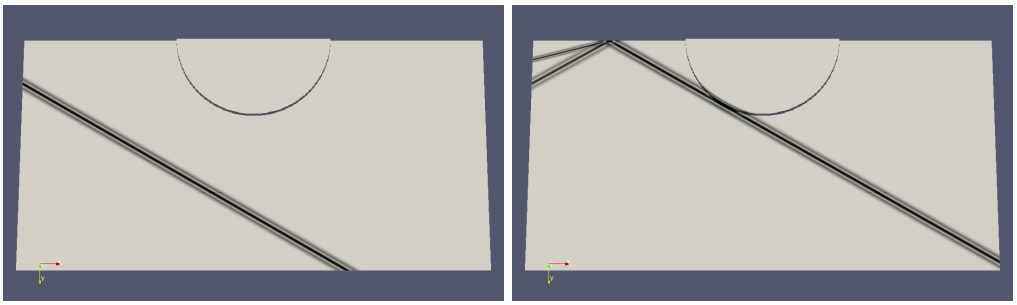


(g) $t = t_7$



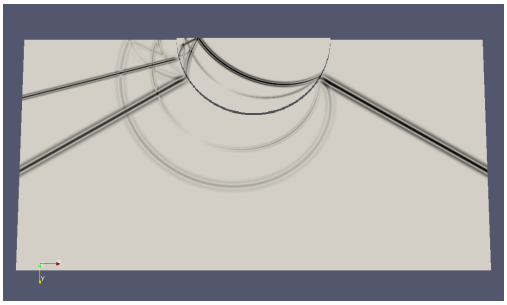
(h) $t = t_8$

Figure 3.48: Snapshots of the SV-Wave incident to a rectangular valley at 10 degrees.

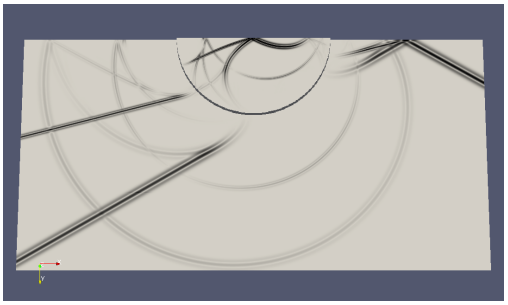


(a) $t = t_1$

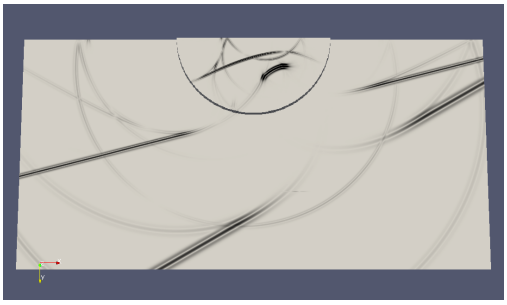
(b) $t = t_2$



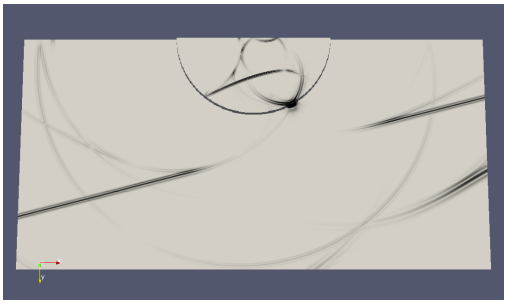
(c) $t = t_3$



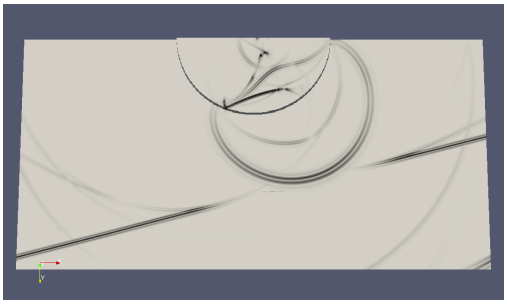
(d) $t = t_4$



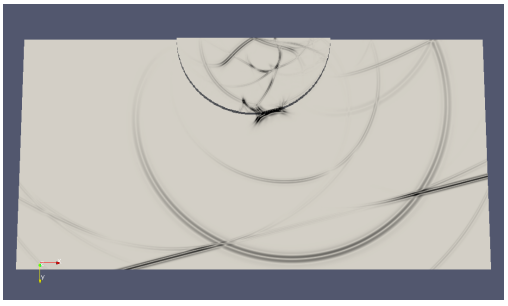
(e) $t = t_5$



(f) $t = t_6$



(g) $t = t_7$



(h) $t = t_8$

Figure 3.49: Snapshots of the P-Wave incident to a rectangular valley at 30 degrees.

Table 3.1 shows the number of DOFs and time for each one of the models. As the wave velocity and geometry are defined the same for incident SV-Waves and P-Waves, the number of DOFs is the same for both waves. Time depends on the number of DOFs and the time simulated, which was 12 seconds. The number of steps for all the models was automatically calculated as 64.000. The tests show that the algorithm can be scaled to use hundreds of computing nodes. Problems of larger sizes would require even more computing nodes but could be in theory scaled. 3 dimensional problems require more DOFs but not more steps as they depend on the side of the elements, (squares or cubes), which does not grow with the dimension, so scaling the problem to 3D would require more computing nodes but not more time.

Table 3.1: Number of DOFs and time for each model.

Model		DOFs	Time
Rectangular Canyon	SV-Wave	51.583.112	6.492s
	P-Wave	51.583.112	6.564s
Semi-Circular Canyon	SV-Wave	51.965.042	7.275s
	P-Wave	51.965.042	6.698s
Rectangular Valley	SV-Wave	58.696.002	7.496s
	P-Wave	58.696.002	7.644s
Semi-Circular Valley	SV-Wave	57.550.134	7.510s
	P-Wave	57.550.134	7.680s

The semi-circular canyon at $f_{peak} = 8$ Hz, $f_{max} = 32$ Hz, was evaluated to know the scalability of the program. Fig. 3.50 shows the times of the solutions when the number of processors is increased from 40 to 240 processors. The curve was fitted to the function $Time = k * \left(\frac{1}{Processors}\right)^n$. Obtaining the values: $k = 62720$ and $n = 0.7449$. Linear scaling, which is what parallel programs seek, would mean $n = 1$.

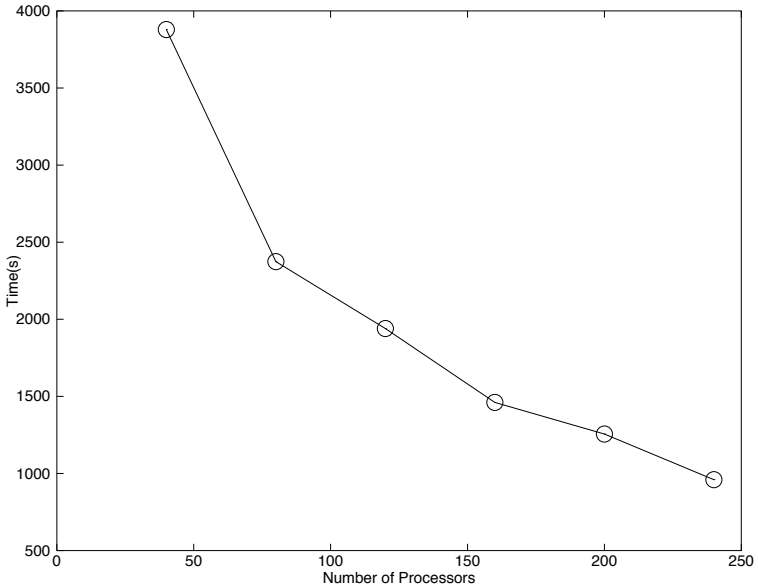


Figure 3.50: Time of the solution of a semi-circular canyon for a different number of processors.

Chapter 4

Interpolation of 2D-cross Sections from Earth Crust Formations: Application to the Construction of the Community Velocity Model for the Aburrá Valley

4.1 Abstract

A Community Velocity Model (CVM), is a computer program and its data files that provide information about earth's elastic properties from the surface to a given depth. CVMs are necessary to accurately predict the outcome of an earthquake using large scale simulations. Aburra's Valley interpretative geological model was created as an starting point of information about earth's formations and is comprised of several 2D slices, each one with different formations delimited by polygons. Interpolating these slices is necessary but there are 3 characteristics that make those slices difficult to interpolate: (i) there are many formations, (ii) slices are far and have a far geometry and (iii) the topology of the slices is very different. We propose an algorithm that is adequate to solve the interpolation problem with the previous constraints and shows good visual results of progression between slices. The model is adequate to create the CVM and was used to simulate earthquakes in Aburra's Valley taking into account the topography of the region.

4.2 Introduction

A Community Velocity Model, CVM, is a computer program and its data files that provide information about earth's elastic properties from the surface to a given depth. Usually the properties are wave velocities and density. CVM's are necessary to accurately predict the outcome of an earthquake using large scale simulations. Scientists routinely use different numerical methods to solve large scale models of earthquakes and compare their solutions using the same CVM model. The term Community refers to the gathering of information from several sources and studies to build an accurate representation of the earth below the surface. One of the few CVMs in existence

is Southern California Earthquake Center's, (SCEC's CVM), which describes the elastic properties of Southern California, [25]. It has information up to 100 Km in depth from inverse tomographic models into one single 3D model, [26, 27, 28]. The CVM is arranged in voxels, cubic boxes that contain the value of a formation, [25]. These voxels are arranged with different precisions with information built from inversion methods and tomographic iterations.

The present work creates a starting version of a Community Velocity Model for Aburra's Valley using the information present in [29]. This model will be used to simulate earthquakes and the effect of topography in exiting numerical methods. The interpretative model created in [29] consists of several images of the earth perpendicular to the surface called slices. This representation is the most used form by geologists that construct interpretative models and is also is used in 2D geological surveys. Interpolation is then required to have a full 3D model where a property of the material is defined everywhere. However, current interpolation methods are not useful for the problem at hand which has three characteristics: (i) there are many formations, (ii) slices are far and have a far geometry and (iii) the topology of the slices is very different. A complete, (to the author's understanding), literature survey that shows the necessity of a new interpolation method that accounts for these 3 constraints, can be found in section 4.3.

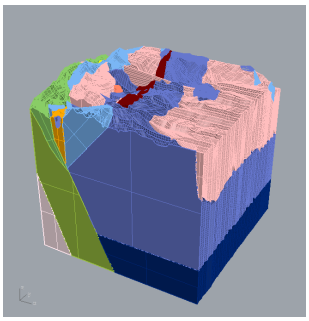
We propose a new method to interpolate slices of earth formations, coming from interpretative models or other models where inter-slice distances are sufficiently large, to create a full 3D model of a region. The paper is divided as follows: (i) we review the methods that have been proposed in several fields to interpolate slices and why they were not appropriate to solve the problem at hand in 4.3, (ii) we explain the rationale behind our approach and propose a two-step algorithm to solve the problem in section 4.4, (iii) we review the results achieved when using the algorithm to interpolate the information in the interpretative model of Aburra's Valley in section 4.5 and (iv) we show our conclusions about the algorithm and its results in section 4.6.

4.3 Review of Methods to Interpolate Slices into a Full 3D Model

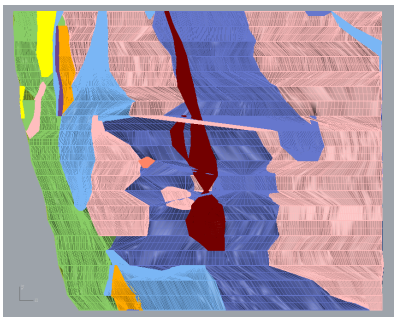
A slice is the image composed by the set of points with a scalar value, each point belongs to a formation. In a slice, many different formations exist. Two different polygons can enclose the same formation in the same slice also. The points are in a plane perpendicular to the surface of the earth, and the given formation is a measure or an expertly guessed value which are difficult to obtain. The slice is delimited by the topography which can be easily obtained with precision. The problem we are studying is obtaining values of formation for points that are between the slices, given slices that are sampled far apart but whose formations are presumed to have continuity. Inter-slice distance for Aburra's Valley interpretative model is 2 Km.

Commercial software sometimes brings features to interpolate slices into a 3D triangulation that can be used for querying points inside regions. Fig. 4.1 is the result of the interpolation made using software Rhino®. This result shows intersections of the triangulations that are incorrect for a CVM. Fig. 4.2 shows a cut made to Rhino's interpolation. It shows that there are empty regions or formations that are also incorrect in the creation of the CVM.

2D Images are usually composed of pixels with values and a first approach is to use voxels delimited by the pixels for slice interpolation. In voxel based methods, the pixels between two slices serve as corners of a volumetric cell and the values interpolated in the voxel depend on them. The most simple methods use linear interpolation [30, 31, 32], which only uses the information in the



(a) Perspective View of the solid interpolated.



(b) Top view of the solid interpolated.

Figure 4.1: Perspective and top views of the solid Modeled with Rhino® which shows intersections

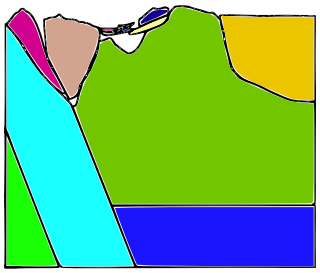


Figure 4.2: Cut of the solid that shows holes

2 nearest oposite vertices of the voxel and trilinear interpolation, [30], which uses information from the 8 vertices of the voxel. Higher order interpolations use near voxels to compute the interpolation of the value with a higher order function, for example tricubic, [33], quadratic, [34], and bicubic, [35]. These methods rely on small transitions from one slice to another. If the same formation does not occur between the two sides of the used voxels, the transition between the two formations is not kept. Another shortcoming of voxel algorithms is that they also rely on the pixel distance to inter slice distance ratio. This assumption is broken when slices are vector images, in which the formations can be sampled with any granularity.

For geophysics and earth resources' exploration there is an entire field that focuses on statistically analyzing and interpolating spatial datasets called geostatistics. In this field, the techniques to interpolate are called Kriging. For a detailed introduction to geostatistics see [36]. Kriging is used for interpolation of slices in [37]. Because it's a statistical tool, it has optimality guarantees. In practice, [37] shows that Kriging slices relies on the information of the voxel or the neighborhood of voxels, as the previous voxel based methods so it's limited when inter slice distance is big.

Shape-based interpolation algorithms are another class of methods, [38, 39]. These algorithms start with a binary image, where the object is one and the outside is 0. Then they convert the image to a grey-scale image based on the distance to the boundary of the object, positive inside and negative outside. An image interpolation method is then used to find the boundaries inter-slices. The problem with this algorithm is its binary nature, only object and outside of object are recognized. It's not possible to have more than one formation.

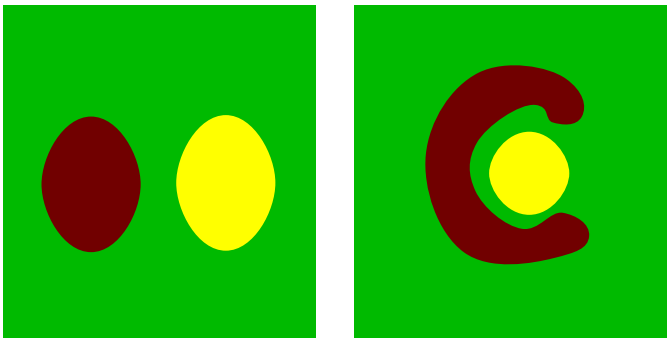
Algorithms that use isosurfaces to interpolate a value between slices are focused on medical images like Magnetic Resonances or Computed Tomographies. The interpolation may be used

to look at the tissues from a different perspective or to have a 3D planning of a surgery. The most used techniques to obtain isosurfaces, (which are the collection of points of a same value within a volume of space), and from those interpolate the value in the region between the slices, is the marching cubes algorithm in [40]. The algorithm is designed for visualization, so incorrect topology is acceptable as long as it's correct enough to visualize. The marching cubes and other reconstruction algorithms have been extended to create topologically correct surfaces, for example in [33] and [41]. Both algorithms deal with the problem of creating a surface which is locally topologically equivalent to a disk in every point, they don't deal with the problem of creating shapes from samples that are topologically equivalent to the original. A complete survey of the marching cubes algorithm was published in [42]. These algorithms also use voxels, but to create isosurfaces. They can only be locally correct and don't give guarantees about the shape so they are not appropriate for big changes in the geometry or topology.

Other kind of techniques to create isosurfaces use pattern recognition techniques and triangulations like Delaunay's, [43, 44, 45]. Unlike previous algorithms which relied on the voxels and discretization, these algorithms can interpolate between very different and separated slices. They also can ensure the topology is respected in cases like branching and merging. Like the shape based interpolation methods, these algorithms rely on a single formation, in this case bone. If more than one formation is interpolated, the triangulations would overlap and form intersections and holes, like in Figs. 4.1 and 4.2.

4.4 Interpolation Technique for Aburrá's Valley CVM

The concept of interpolating a value from a set of measures is closely related to the general problem of shape reconstruction. Generally, topology and geometry are used as indicators for correctness of an algorithm since [46, 47]. A complete treatment of computational topology and its application to shape and surface reconstruction was published in [48]. A shape is said to be well reconstructed from sample points if it's homeomorphic to the original sampled shape and geometrically is as close as possible. Let S_A and S_B be two geometric shapes, S_A is homeomorphic to S_B if there is a function $f : S_A \to S_B$ such that f is bijective and f and its inverse are continuous. Intuitively, two shapes are homeomorphic if one can be created from another by stretching and bending but not by cutting or glueing. Fig. 4.3 shows two homeomorphic shapes.

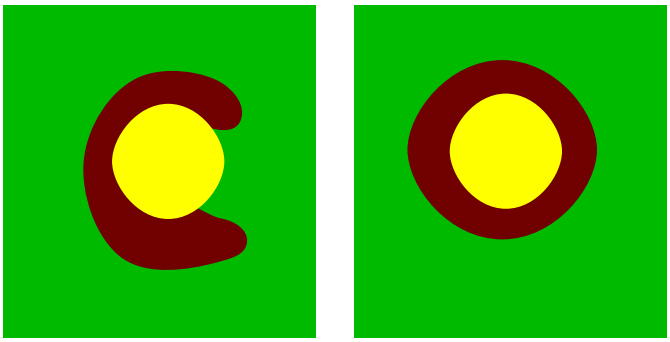


(a) Shape S_A which is a slice with 3 formations.

(b) Shape S_B which is homeomorphic to S_A.

Figure 4.3: Homeomorphic shapes.

If two shapes are homeomorphic it means that they might be slightly deformed until they are converted into one another. If two slices are homeomorphic it means that isosurfaces can be constructed between them that connect every loop to another loop in the adjacent surface. Otherwise, by definition, connecting two shapes is mathematically impossible without a topological transition, because at some point the geometry would have to be glued or cut. That's why commercial software reconstruction of slices fails in such cases, as in Fig. 4.2. However, we can see that even when topology can change, the transition can be simple, as in Fig. 4.4.



(a) Shape S_C which is not homeomorphic to S_B but is a transformation of it.

(b) Shape S_D which is not homeomorphic to S_B but is a transformation of it.

Figure 4.4: The shapes are not homeomorphic to those of Fig. 4.3, but they can be transformations from them.

Unlike triangulations that can overlap, leave holes or even have local topological inconsistencies, real distance functions are well defined at every point of the space. That's the reason why they are to chose with problems that, unlike visualization, require a perfect topology. Delaunay's triangulation is possibly the most important geometrical data structure. It's dual is Voronoi's diagram which is the set regions that is closer to each point, (called site), than to any other point. In computational geometry, Delaunay's triangulation is the best definition of closeness available. Proof of this is that Delaunay's triangulation contains the euclidean minimum spanning tree, the relative neighborhood graph, and the nearest neighbor graph. For a complete description see [49]. Delaunay's triangulation, with the appropriate topological transition, is used in many isosurface interpolation papers as main resource. Unlike those papers, we use Delaunay Triangulation to find which polygons of two different slices match. After that matching, we use a real distance function that follows the direction of the polygons. The real distance function is defined at every point between the slices and can be compared to the same distance function to other matching polygons.

Given a triangulation, T, two polygons match according to their connected length. A polygon is denoted as $P = \{L_1, L_2, \ldots, L_N\}$: a set of loops, interior and exterior that encloses a formation, denoted as $Formation(P)$. The formation consists of a certain number of rock strata that have comparable lithology, facies or other similar properties. The loop is a closed line $L = \{p_1, p_2, \ldots, p_N\}$ where $(p_1, p_2), (p_2, p_3), \ldots, (p_N, p_1)$ are segments. The connected length of two polygons with the same formation, denoted as $ConLen(P_1, P_2)$, is the sum of the length of the segments a loop that belongs to P_1 which are part of a triangle of the triangulation that has its opposite vertex in P_2. Algorithm 5 shows the polygon matching scheme.

Data: Two slices, I_A, I_B
Result: $Matched$, a set of pairs of polygons in opposite slices that match
Obtain the Delaunay Triangulation T of both slices;
for $P_1, P_2 : ConLen(P_1, P_2) \neq \emptyset$ **do**
  if $\forall P_J \in I_A$, $ConLen(P_J, P_2) < ConLen(P_1, P_2) \wedge$
    $\forall P_K \in I_B$, $ConLen(P_1, P_K) < ConLen(P_1, P_2)$ **then**
    $MaxLen(P_1, P_2) \leftarrow ConLen(P_1, P_2)$
  end
end
Let $Matched \leftarrow \{(P_1, P_2) : MaxLen(P_1, P_2) \neq \emptyset\}$;
Let $NotMatched \leftarrow \{P_1 : \forall P_2,\ MaxLen(P_1, P_2) = \emptyset\}$;
for *Every polygon* P_1 *in* $NotMatched$ **do**
  if $\exists P_2 \mid ConLen(P_1, P_2) > 0$ **then**
    Let $P_2 \leftarrow$ the polygon with maximum connected length to P_1;
    Let $Matched \leftarrow Match \cup \{(P_1, P_2)\}$;
    $NotMatched \leftarrow NotMatched - \{P_1\}$;
  end
end
for $P_1 \in NotMatched$ **do**
  Let P_2 be the matched neighbor that shares more arc length with P_1;
  Add P_1 to P_2;
end

Algorithm 5: Find a match for every polygon

The loop pairing can be a costly operation so it's not done at the same time that the CVM is run. Preprocessing is done between every consecutive pair of slices and saved independently. That is, if slice I_B has some loops merged when matching with slice I_A, those same loops are not merged when matching I_B and I_C, the loops start from scratch. The information saved by the preprocessing algorithm is the set of loops and matches in I_A and I_B after the algorithm has finished.

The polygon matching is created to find a direction in which two polygons that are distant are related. After matching the polygons, a vector $\vec{v}$ is found as $\vec{v} = c_1 - c_2$, the difference between the centroids of polygons P_1 and P_2. For any point p we find the intersection between $p + \vec{v}$ and both slices. The distances to the polygons are then weighted according to the distance to the slices. When the weighted distances to two formations are the same, there is not a definition about which should be chosen. As an heuristic, we used the smallest one. For speed when querying, the directed vector between the centroids of the matching loops is also calculated during the preprocessing stage.

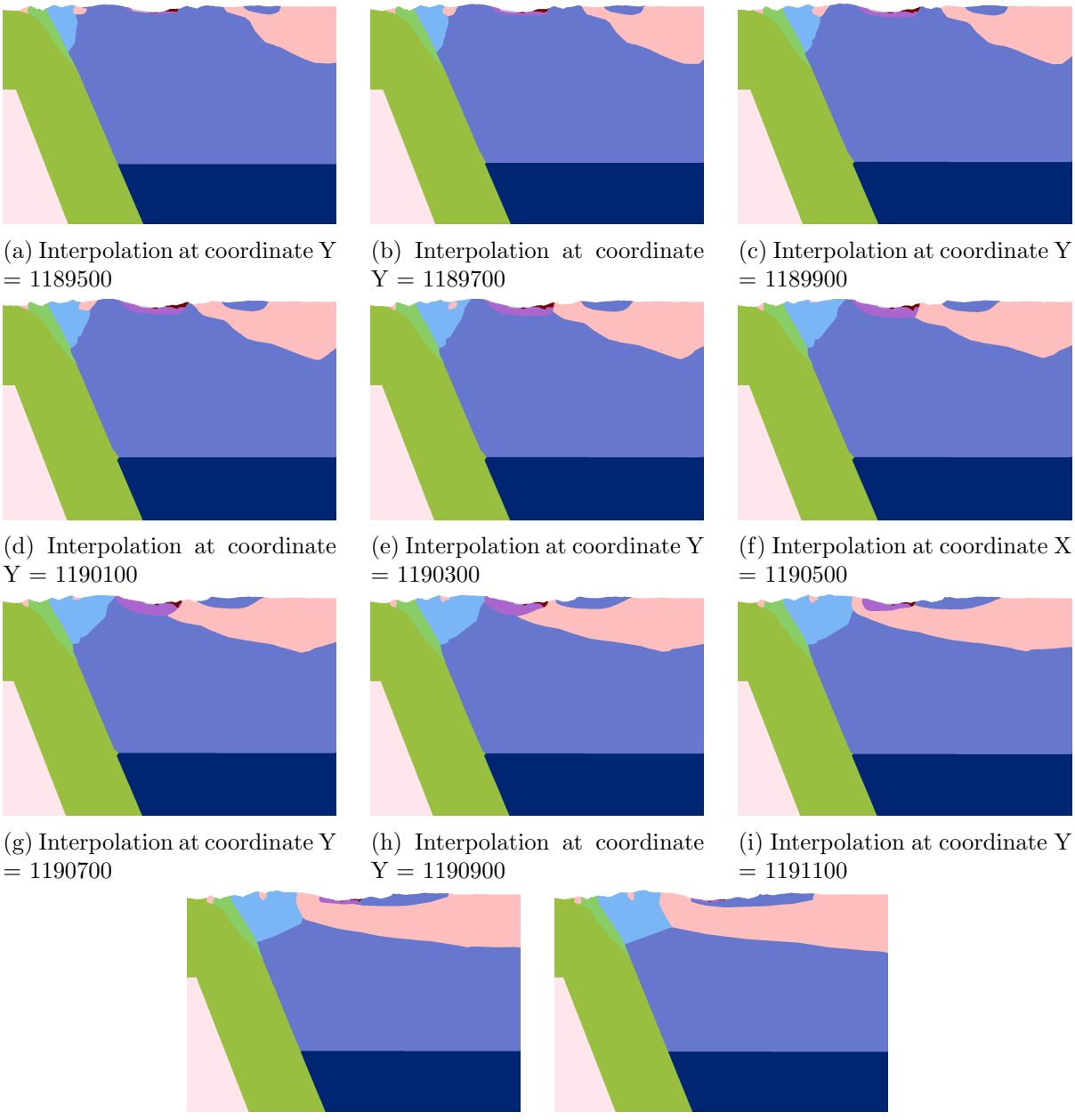
Data: A point p with three spacial coordinates, p_x, p_y, p_z
Result: The formation $Formation(p)$ in which that coordinate lies
if *p is below the topography* **then**
  Let $Formation(p) \leftarrow air$.
end
Let I_A, I_B be the pair of consecutive slices such that the point lies between them;
for $(P_1, P_2) \in Matched$ **do**
  Let $\vec{v}$ be the vector between the centroids of P_1 and P_2. Let q_A be the intersection between the plane that contains I_A and $p + \vec{v}$;
  Let q_B be the intersection between the plane that contains I_B and $p + \vec{v}$;
  if q_A *inside* P_1 **then**
    Let $d_1 \leftarrow 0$;
  else
    Let d_1 be the distance to P_1;
  end
  if q_B *inside* P_2 **then**
    Let $d_2 \leftarrow 0$;
  else
    Let d_2 be the distance to P_2;
  end
  Let $Distance(P_1, P_2) \leftarrow \left(1 - \frac{\|q_A - p\|}{\|q_A - q_B\|}\right) d_1 + \left(1 - \frac{\|q_B - p\|}{\|q_A - q_B\|}\right) d_2$;
end
Let $Formation(p) \leftarrow \min Distance(P_1, P_2)$;

Algorithm 6: Find a formation given a point inside the interpolated region

Wave velocities are the result of combining the formation in with geophysical considerations which are outside of the scope of this paper.

4.5 Results

The algorithm was designed and tested to create the CVM of Aburrá's Valley, that comprises the city of Medellín and other smaller urban centers. The 2D slices used to build the CVM of Aburrá's Valley come from information gathered in [29]. It was gathered by the authors using cartographical, geological and geophysical information of studies made in the region. The information is compiled in cuts every 2Km up to 30 Km in depth. The rest of the information contains superficial geological formations and the elevation of the mountains. Aburrá's Valley is a very mountainous region with heights up to 3200 m.a.s.l, even when its official, (mean), elevation is 1479 m.a.s.l. This makes it very different from Southern California's region, which can be flattened without losing a lot of detail during the simulation. The model will be used to simulate seismic events caused by Cauca-Romeral's fault, one of the most important in Colombia, that has ramifications in the region modeled. The model will allow not only the simulation of accurate seismic events in the region but also allow the development of numerical methods that take into account the topography. Fig. 4.5 shows the progression between slices at coordinates Y = 1189500 and Y = 1191500. The topology changes in the formations several times until one shape is deformed into another, which was not possible to achieve with previous interpolation methods.



(a) Interpolation at coordinate Y = 1189500

(b) Interpolation at coordinate Y = 1189700

(c) Interpolation at coordinate Y = 1189900

(d) Interpolation at coordinate Y = 1190100

(e) Interpolation at coordinate Y = 1190300

(f) Interpolation at coordinate X = 1190500

(g) Interpolation at coordinate Y = 1190700

(h) Interpolation at coordinate Y = 1190900

(i) Interpolation at coordinate Y = 1191100

(j) Interpolation at coordinate Y = 1191300

(k) Interpolation at coordinate Y = 1191500

Figure 4.5: Progression between slices at coordinates Y = 1189500 and Y = 1191500

4.6 Conclusions and Future Work

We presented and solved the problem of interpolating the information of an interpretative model for Aburra's Valley into a Community Velocity Model. The resultant algorithm is very simple and fast for formation querying. It can be parallelized, however as it's used as a an input to a mesher, it needs to be run by several processes independently and it's better to keep it single threaded. This CVM is very useful to create more complicated ones from tomographic models, as SCEC's, which makes it a very important starting point for having an accurate and complete description of the earth below the surface in Aburra's Valley. The algorithm is very general, so it can be used with models of other regions to create simple CVMs which allows them to comprehend seismic events easily.

As a main conclusion it was shown that it's possible to formulate an algorithm to interpolate slices into a full 3D model that can be used with an automatic mesher, instead of a human interpreter as in the case of algorithms for visualization. The algorithm can interpolate changing topology, distant geometry and several formations. Previous algorithms could only solve two of the three problems at the same time.

As a future work, new weighting functions that also are defined for every point in the space can be defined and the method generalized. The most important future work should be giving topological guarantees for slices with different topologies and large geometrical distances.

Chapter 5

Numerical Simulation of the Seismic Response of the Aburrá Valley Subject Under the Incidence of Plane Waves

The main objective of having a software that can create large scale simulations is being able to simulate realistic problems. In this case we simulated an SV-Wave with incident angle 0 hitting a slice of Aburra's valley at a certain latitude. SV-waves were used as incoming waves as they are the most disastrous of them. It is expected that further analysis and use of the work presented here will provide a deeper understanding about what could happen to structures under a seismic event. However, an analysis of the scattering patterns, transfer functions or more applied, vulnerabilities of Aburra's Valley is out of the scope of this work and only snapshots will be presented.

Chapter 4 shows a complete explanation of how Aburra's Valley CVM was obtained to be able to create this model. The model was then created using the CVM at several latitudes bringing back the information in it. The coordinates of the CVM are the same described in [29], which is the input for our analysis. Appendix B describes the input of the CVM's algorithm and has images that detail how coordinates are located in the region of interest.

The simulations were made using Ricker pulse with peak frequency $f_{peak} = 2$Hz and maximum frequency $f_{max} = 8$Hz. The material properties used are in table 5.1 that shows each name of a formation, its color and its elastic properties, P-wave velocity (α), Poisson's module (v) and density (ρ). The materials are elastic with no damping. Plane waves come from a source at infinity which is modeled as a uniform half space. As described in section 2.3, the Domain Reduction Method used to model the incoming waves needs to place the region of interest into another region separated by a strip of elements where the displacements from the original model are applied. As the plane waves modeled respond to a half-space, the outside region has to be modeled with a single material property hardest material of the model, which is the one with P-wave velocity 4500. It can be said that the model is set-in in this homogeneous region.

This chapter shows the result for slice at Y=1170000, described in Fig. 5.1 shows the slice with the different colors related in 5.1. Figs. 5.2 and 5.3 show snapshots of the magnitude of the displacements at 16 time frames. The wave velocity of a region is shown by extruding the region with lower wave velocity the most: the more extruded a region seems, the less wave velocity it has.

Table 5.1: Properties table for Aburra's Valley.

Name of the formation (Spanish)	Color	P-Wave Velocity (α)	Poisson's Module (v)	Density (ρ)
Esquistos Cuarzo-Sericíticos y Cloríticos		2000	0.25	2500
Anfibolitas		4500	0.25	2900
Aluviales		400	0.25	2400
Cuerpos Plutónicos Cretácicos		4500	0.25	2200
Secuencia Volcánica		1500	0.25	2800
Gneises Sintectónicos		3100	0.25	2700
Complejo Arquía		4500	0.25	2900
Barroso		1500	0.25	2800
Granulita		4500	0.25	3100
Metagabros		4500	0.25	3000
Rocas Ultramárficas		4500	0.25	1270
Neis de la Iguaná		3100	0.25	2700
Gabros asociados al SFCR		4500	0.25	3000
Sedimentos Terciarios		400	0.25	2400
Depsito de Vertiente		400	0.25	2400
Migmatitas, Granulitas y Gneises Anfibolíticos		400	0.25	2400

The simulation generated a mesh with 14.421.706 DOFs and 77610 steps. The simulation ran for 41.388 seconds on 80 Intel(R) Xeon(R) CPU E5430 at 2.66GHz.

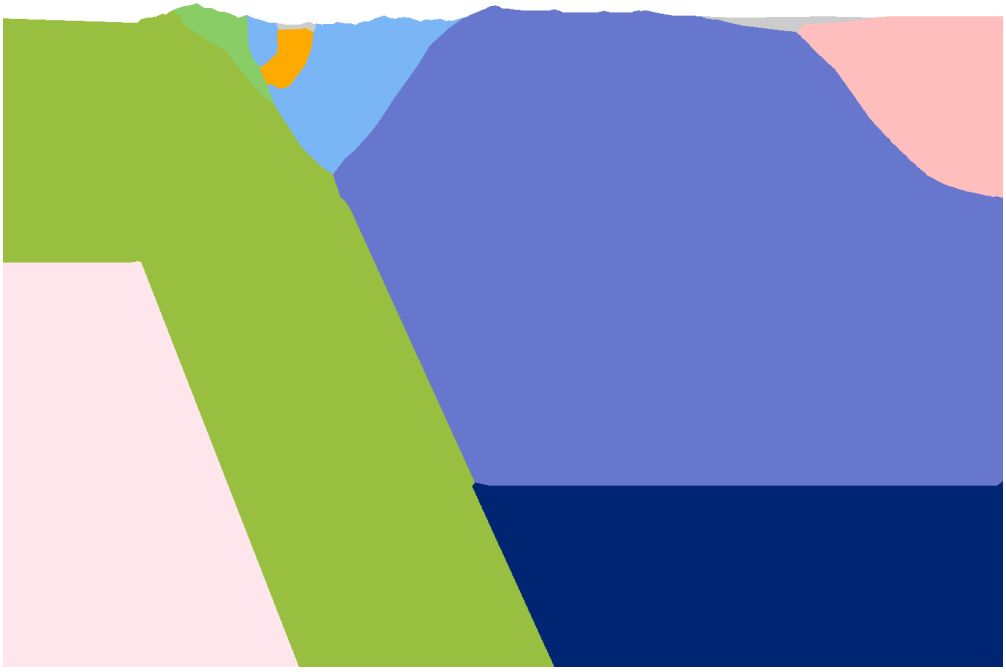
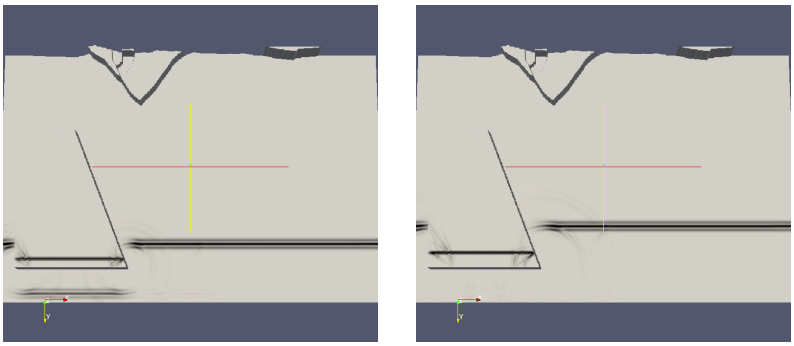
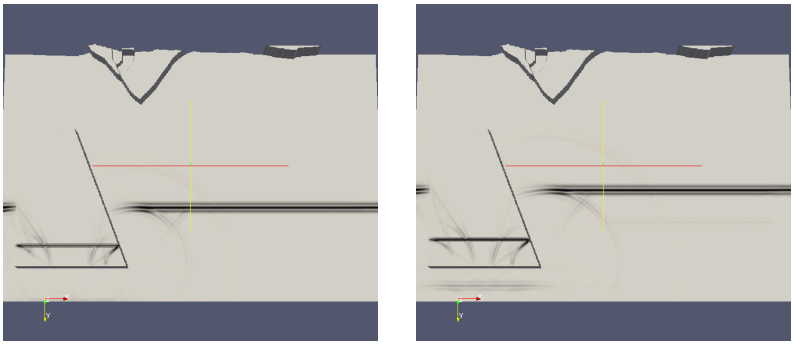


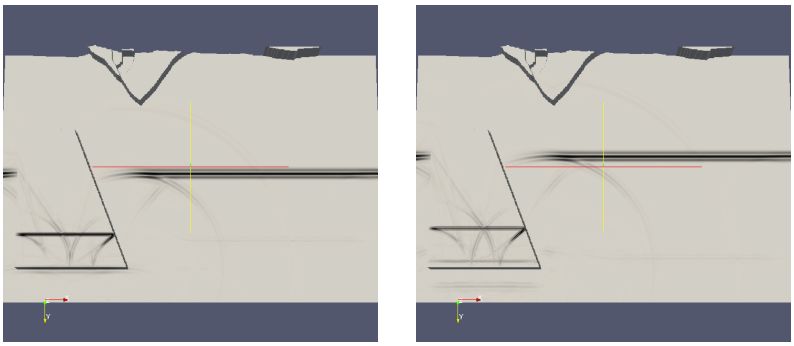
Figure 5.1: Slice of Aburra's Valley at Y=1170000.



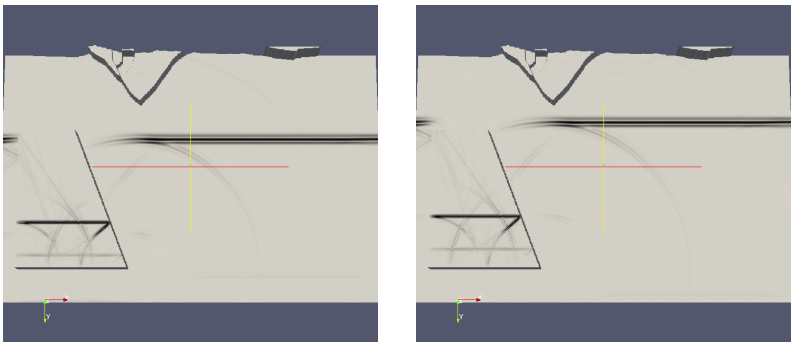
(a) $t = t_1$ (b) $t = t_2$



(c) $t = t_3$ (d) $t = t_4$

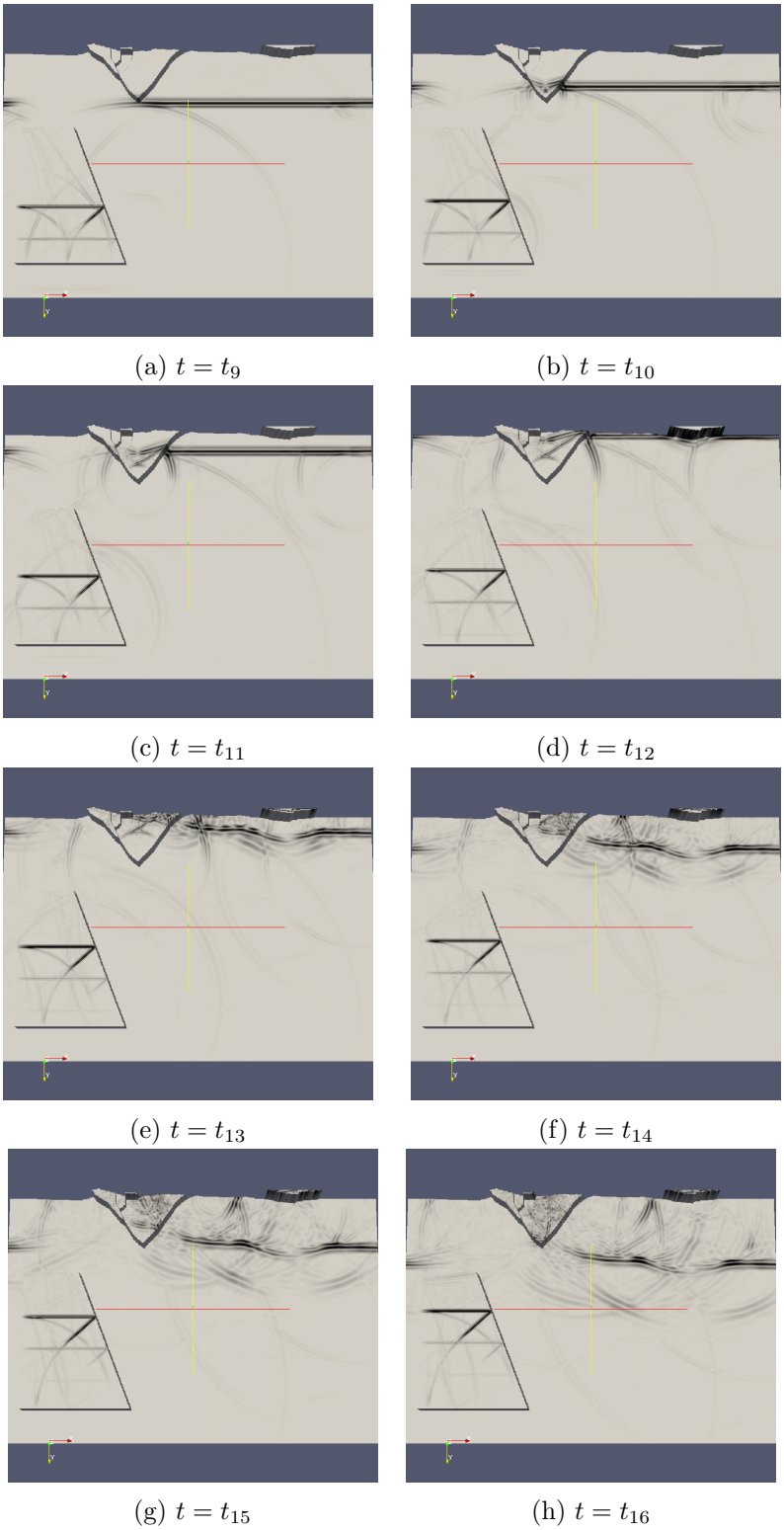


(e) $t = t_5$ (f) $t = t_6$



(g) $t = t_7$ (h) $t = t_8$

Figure 5.2: Snapshots of an SV-wave incident to Aburraś Valley at 0 degrees, Y coordinate = 1170000. t_1 to t_8



(a) $t = t_9$ (b) $t = t_{10}$

(c) $t = t_{11}$ (d) $t = t_{12}$

(e) $t = t_{13}$ (f) $t = t_{14}$

(g) $t = t_{15}$ (h) $t = t_{16}$

Figure 5.3: Snapshots of an SV-wave incident to Aburraś Valley at 0 degrees, Y coordinate = 1170000. t_9 to t_{16}

Chapter 6

Conclusions and Future Work

The aim of this thesis was to compile all the necessary software and experiences that we had doing the project that wanted to appropriate large scale simulations of wave propagation problems and use them to model Aburra's Valley. DAMIAN-PAR and Aburra's Valley CVM where created to fulfill this need, and this thesis tried to discuss the different challenges we were faced with and how we solved them.

The main conclusion of this thesis is that the software required to simulate large scale wave propagation problems in earthquake engineering needs to be carefully crafted to take the biggest possible advantage of the computational resources and leave the user with only the duty to model, run and analyze the problems without having to be worried about all the complexities involved. DAMIAN-PAR is an all-in-one software in which the analyst only needs to specify the relevant information to the problem and the solution is returned in formats which can be processed further in his personal computer. No design of the mesh, no manual application of loads and no real mesh rendering are required. This is because all these tasks are very difficult to do manually in large scale simulations where there are millions to billions of elements and the analyst would struggle just imagining that number.

Similarly, a new interpolation algorithm to model Aburra's Valley CVM was invented because it solved problems like holes and intersections. These problems are not usually tackled in visualization algorithms because the user could quickly intervene when analyzing; however, as DAMIAN-PAR and other large scale simulation software are automatic software with numerical methods that could fail, the problems became very relevant, pushing us to create algorithms robust enough to tackle most inputs.

The result of the work written here will help EAFIT and others to appropriate the key large scale simulation technologies that are helping scientists everywhere to understand complex phenomena better. As part of a number of projects that use APOLO, large scale simulation of wave propagation problems in earthquake engineering has helped EAFIT to become a university capable of pushing the boundaries of science by solving problems which were impossible or impractical to solve before APOLO. Specific conclusions are shown in the following sections.

6.1 Conclusions to DAMIAN-PAR design and validation

Everything in DAMIAN-PAR was designed around the quadtree Finite Element Method scheme. As such, the choice of a numerical method is the single most important task when designing software

for large scale simulations. The quadtree provided answers to several of our challenges, like how to design the pre-processor in a way that was efficient and without bottlenecks, what was the most appropriate scheme for modeling the incoming waves or how to present the solution to the analyst in a way that did not create more challenges.

The core of this thesis was then the validations. The validations provided an opportunity that the researchers that proposed the technologies used in DAMIAN-PAR did not have: to really test the methods against problems that are well understood by the community evaluating every single source of error associated with an approximation. Given the approximations, it was expected that the semi-circular canyon and the semi-circular valley presented a few problems while the rectangular canyon and the rectangular valley could be modeled perfectly. Instead, we found that the semi-circular canyon had problems as expected, but also the rectangular valley had problems.

The rectangular valley was one of the simplest geometries in terms of what a quadtree mesh could represent, however, it was the most complex model in wave propagation problem terms. The rectangular valley has four sources of diffraction that generate new waves each time the wave is reflected and refracted at these points. Waves inside the rectangular valley remain there for a long time bouncing until they dissipate completely. We have to further investigate if the problems we saw in the rectangular valley occurred because of the quadtree discretization. If that's the case, the method has to be modified to be able to solve such problems.

As a future work there are several areas that need further research:

1. The absorbing boundaries used are the simplest found in the literature and several evolutions of them have been proposed by other researchers. Absorbing boundaries that in the literature were supposed to be better were not used because most of them were created for the frequency domain and, to the best of our understanding, absorbing boundaries have not been created specifically for large scale simulations.

2. The octree has been tested against layered half-space problems where the wave velocity of the layer is half of the half-space wave velocity. Also our validations always used wave velocities for the soft material which were half of the hard material's ones. Validations where the soft material has a wave velocity of a fourth or a tenth of the wave velocity of the hardest one have not been done. This would require the quadtree to create a progression of squares of different sizes at the interface, which exists in Aburra's Valley and its supported but has not been validated.

3. The topography problems of the quadtree have already been solved by other researchers but not in DAMIAN-PAR. The effect of creating elements that approximate the topography correctly has to be evaluated.

6.2 Conclusions to Aburra's Valley CVM

Aburra's Valley CVM created a starting model using new interpolation method that did not exist. The current CVM will be the base for further studies which take into account historical earthquakes, information about the epidermis from holes and geophysical considerations; and build a model that is more detailed. Even when it requires a lot of work to become a realistic model of Aburra's Valley, the CVM has worked for DAMIAN-PAR and has also been used by other researchers to solve wave propagation problems.

The algorithm created has shown to be very powerful and a perfect match to the problem at hand and is very general, so it can be used with models of other regions to create simple CVMs which allows researchers to comprehend seismic events there. It was shown that it's possible to formulate an algorithm to interpolate slices into a full 3D model that can be used with an automatic mesher, instead of a human interpreter as in the case of algorithms for visualization. The algorithm can interpolate changing topology, distant geometry and several formations. Previous algorithms could only solve two of the three problems at the same time.

6.3 Conclusions to the Simulation of Aburra's Valley

The simulation of Aburra's Valley was not the core of this thesis but creating a detailed analysis of it is certainly a work necessary to increase our understanding of earthquakes in the region, define action plans given catastrophic events, and reduce our vulnerability by creating better civil structures. The most important thing we could show simulating Aburra's Valley was that it was possible to solve large scale wave propagation problems with heterogenous material properties and complex topography so a future project can make a detailed analysis.

Appendix A

Usage of DAMIAN-PAR

A.1 Folders and Files

DAMIAN-PAR has the following files and folder structure:

The folder hostscripts contains scripts specifics for the different machines that will run the exfem application. They make easy to prepare the environment for running the application in apolo or other clusters.

```
|-- hostscripts
   |-- qapolo.sh
   \-- qcoates.sh
```

Libs contains the libraries required for the application to run. They have to be compiled for the operating environment of the cluster.

```
|-- libs
   |-- libacml_mv.so
   |-- libacml.so
   |-- libmetis.so
   \-- libparmetis.so
```

All the models are set in folders which have a problem.in file inside. This file defines all the necessary parameters of the given problem.

```
|-- circul
   \-- problem.in
|-- cvm
   \-- problem.in
|-- rectani45
   \-- problem.in
|-- rectani60
   \-- problem.in
|-- valley
   \-- problem.in
```

The source files have extensions like .f90, .cpp, .h or .m4. They contain the source code of the solution. The source files are built with the command make.

```
|-- comm_temps.f90
|-- comm_temps.m4
|-- communicator.f90
|-- explicit.f90
|-- formulation.f90
|-- gdb-exts
|-- io.cpp
|-- io.h
|-- io_utils.f90
|-- main.f90
|-- Makefile
|-- mesh_utils.f90
|-- model.f90
|-- quadtree.f90
|-- test.f90
|-- utilities.f90
|-- util_temps.f90
|-- util_temps.m4
\-- visualization.f90
```

The environ.in file contains information necessary to run a simulation in the environment of the cluster that is being used. It points to the temporary folders in each computer that will store the solution of the problem, and the shared folder where the final visualization files will be exported.

```
|-- environ.in
```

The exfem file is the executable of the DAMIAN-PAR application. It is compiled to run with mpirun in several machines.

```
\-- exfem
```

To begin the configuration of the problem in the environment, the environ.in file must be modified. The environ.in file looks like this:

```
ENVIRONMENT FILE EXFEM
VISUALIZATION:
/home/rserrano/output
TMP:
/state/partition1/tmp
CVM PATH:
/home/rserrano/slicerecon
```

The file starts with the header, which corresponds to the text: "ENVIRONMENT FILE EXFEM". The following fields are required by DAMIAN-PAR :

1. VISUALIZATION: It's the path where a folder for the output files for visualization will be written.

2. TMP: It's the folder where temporary files will be created to store the solution to be queried and visualized later.

3. CVM PATH: It's the folder where the Community Velocity Model executable is located.

After that, the problem.in file must be configured. For that, a folder named as the user wants to name the problem must be created. For example, the folder problem1 can be created and a problem.in file must be placed in it. The contents of the problem.in file are the following:

```
PROBLEM FILE EXFEM
DOMAIN:
16000 10000
TIME:
20 0.1
CH PERIOD:
0.125
MATERIALS:
1
p 1000 v 0.33 1000
PROBLEM:
WAVEP
9 12000 8000 8000 0 1 14 1.04719755 0.1 0.010416666666666
GEOMETRY:
RECTAN
4 8000 0 1000 2000
```

The file starts with the header, which corresponds to the text: "PROBLEM FILE EXFEM". The following fields are required for the problem to run:

1. DOMAIN: It's the dimensions of the initial rectangle that will compose the domain of the problem. It received two parameters: width and height. In the case of the example, the domain will be a rectangle of dimensions 16000m $\times$ 1000m.

2. TIME: Controls the amount of time that the problem will run. It also controls how often the solution should be saved to the file. It contains 2 parameters: total time and the time between each solution storage. In the case of the example, the total time of the simulation will be 10s. The solution will be save each 0.1s, about 200 times.

3. CH PERIOD: It's the characteristic period of the problem: $T_{ch} = \frac{1}{f_{ch}}$. It will control the mesh and time-step of the problem. In this case, the characteristic period is 0.125s, the characteristic frequency would be 8Hz.

4. MATERIALS: It contains the material properties of the different materials that will be found in the geometry. Depending on the chosen geometry, a certain number of materials will be required. MATERIALS is followed by a single number N in the following line. N is the

number of materials that will be used in the problem. The following N lines will contain the different material properties required. Each line consists of 5 fields. The first is the name of the first property. Properties can be:

(a) p: the p-wave velocity or α.

(b) s: the s-wave velocity or β.

(c) v: the Poisson's ratio.

(d) E: Young's modulus.

(e) K: Bulk modulus.

(f) l: first Lamé parameter.

(g) G: Shear modulus or second Lamé parameter.

The second field contains the value of that property. The third field contains the name of the second property, as in the previous list. The fourth field contains the value of that property. The fifth field contains the density, (which is used regardless of the other properties). In this case, we have 1 material, that has a p-wave velocity of 1000m/s, a Poisson's ratio of 0.33 and a density of 1000Kg/m^2

5. PROBLEM: It contains the definition of the input source that will be simulated. The first line contains the kind of the problem to be simulated. The second line contains the parameters for the input source. It begins with an integer N, the number of parameters to read. It is followed by N reals with the required parameters. Up to the date of the report, 3 kinds of input sources have been fully implemented and tested:

   (a) WAVEP: It implements an input plane P-wave convoluted with a Ricker pulse. The following parameters are required: the number of parameters, width of wave (approximately), height of wave, (approximately), X coordinate of origin, Y coordinate of origin, index of homogeneous material of the wave, time to origin, angle, amplitude and sigma of Ricker's pulse. As the mesh is automatic, the positions of of the box enclosing the input wave will be approximately the described in the parameters of the input file.

   (b) POINTN: It implements a punctual force (Neumann's Boundary Condition) in the positive Z direction. The following parameters are required: the number of parameters, amplitude, sigma of Ricker's pulse, time to start, maximum distance from points to the nodes that will have the load applied, X coord, Y coord, X coord, Y coord, ... The number of X coord, Y coord points will be the number of parameters less four. As the mesh is automatic, the positions of the nodes where the load will be applied will be the ones inside the circles described by the points and the distance in the input file.

   (c) POINTD: It implements a punctual displacement (Diriclet's Boundary Condition) in the positive Z direction. The following parameters are required: the number of parameters, amplitude, sigma of Ricker's pulse, time to start, maximum distance from points to the nodes that will have the load applied, X coord, Y coord, X coord, Y coord, ... The number of X coord, Y coord points will be the number of parameters less four. As the mesh is automatic, the positions of the nodes where the load will be applied will be the ones inside the circles described by the points and the distance in the input file.

6. GEOMETRY: It contains the definition of the geometry of the problem. It may be a classic problem, like the semicircular valley, or the community velocity model. The second line contains the parameters of the geometry. It begins with an integer N, the number of parameters to read. It is followed by N reals with the required parameters. Up to date, 6 kinds of geometries have been fully implemented and tested:

   (a) VALLEY: It's a sedimentary semicircular valley. It requires 2 material definitions to work. It needs the following parameters: the number of parameters, the center of the semicircle, and the radio of the semicircle. The representation of the circle is approximated with squares.

   (b) CANYON: It's a semicircular canyon. It requires 1 material definition to work. It needs the following parameters: the number of parameters, the center of the semicircle, and the radio of the semicircle. The representation of the circle is approximated with squares.

   (c) RECTAN: It's a rectangular canyon. It requires 1 material definition to work. It needs the following parameters: the number of parameters, the center of the rectangle, the X distance from the center to be carved and the Y distance from the center to be carved.

   (d) CVMPLX: It constructs the geometric model from the information, (materials and carving), of Medellin's Community Velocity Model. It creates a two dimensional model from the information in a plane perpendicular to the X coordinate. The parameters required are: the number of parameters, the X coordinate of the plane, the Y coordinate corresponding to the 0.0 X coordinate of the model, the Z coordinate corresponding to the 0.0 Z coordinate of the model. According to the common seismic coordinates, the Z is positive in direction to the ground, but the CVM model defines Z as meters above the sea level. The more the Z coordinate increases in DAMIAN-PAR 's model, the more it decreases in CVM's model.

   (e) CVMPLY: It constructs the geometric model from the information, (materials and carving), of Medellin's Community Velocity Model. It creates a two dimensional model from the information in a plane perpendicular to the Y coordinate. The parameters required are: the number of parameters, the X coordinate corresponding to the 0.0 X coordinate of the model, the Y coordinate of the plane, the Z coordinate corresponding to the 0.0 Z coordinate of the model. According to the common seismic coordinates, the Z is positive in direction to the ground, but the CVM model defines Z as meters above the sea level. The more the Z coordinate increases in DAMIAN-PAR 's model, the more it decreases in CVM's model.

   (f) CVMPLZ: It constructs the geometric model from the information, (materials and carving), of Medellin's Community Velocity Model. It creates a two dimensional model from the information in a plane perpendicular to the Y coordinate. The parameters required are: the number of parameters, the X coordinate corresponding to the 0.0 X coordinate of the model, the Y coordinate corresponding to the 0.0 Z coordinate of the model and the Z coordinate corresponding of the plane.

The model is solved running the command the command:

```
mpirun -np # -machinefile file ./exfem solve problem1
```

The application then reads the environment.in file located in the current directory and the problem.in file located in ./problem1/problem.in, creates and distributes the mesh and solves the problem leaving the files containing the solution sparse in the computers that ran the simulation.

After the solution is found, the following command can be run:

```
mpirun -np # -machinefile file ./exfem visualize problem1
```

The application then asks five parameters:

1. xmin: the start of the X coordinates of the model that the user wants to visualize.
2. xmax: the end of the X coordinates of the model that the user wants to visualize.
3. ymin: the start of the Y coordinates of the model that the user wants to visualize.
4. ymax: the end of the Y coordinates of the model that the user wants to visualize.
5. dist: the distance between points that the user wants to visualize.

The program queries and finds the value of the field at the rectangle delimited by the coordinates given and with the precision given by the distance provided, for every time step that is saved in the solution files. It saves them to a set of ParaView exfemXXXX.vtu files in the visualization folder for each timestep. It also creates a file called mater.vtu that contains the material properties of the model at the same rectangle and points: number of material property and p-wave velocity. Finally, it creates a file with Matlab matrices to create a visualization of the sheet.

To make easier the usage in different clusters, the folder hostscripts contains a set of scripts to help to run the model easier. The apolo.sh script runs the code in the apolo cluster in a way that is easier to use. It uses environment variables to define the model and the visualization parameters. It also creates the necessary folders or deletes the existing solution of the problem already in the cluster making it easier to launch several problems. The script is run as follows:

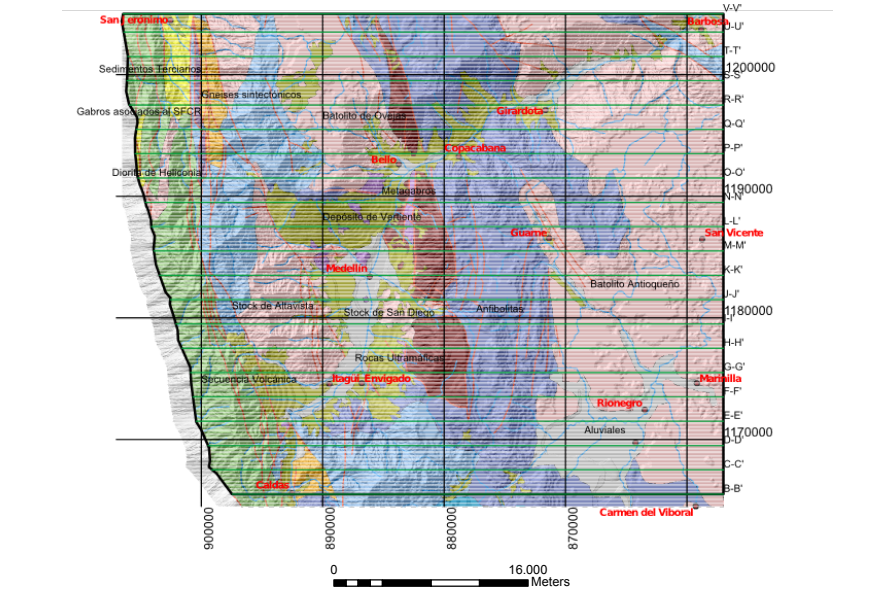
```
MODEL="problem" GEOM1="xmin xmax ymin ymax dist" \
GEOM2="xmin xmax ymin ymax dist" \
GEOM3="xmin xmax ymin ymax dist" \
nohup ./hostscripts/apolo.sh &
```

The application runs the model specified by the environment variable MODEL and leaves a set of .zip files in the visualization folder for each of the visualizations specified in the GEOM1, GEOM2, GEOM3, ..., environment variables. Each .zip file contains the .vtu files generated for that visualization number.

Appendix B

Input information from Acevedo's Survey

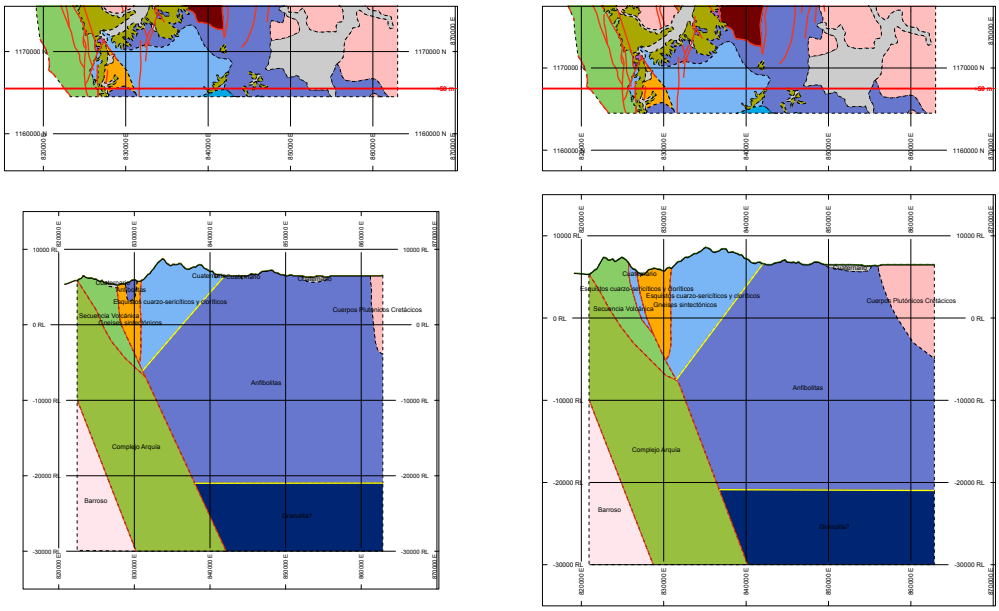
The slices that serve as an input to this project CVM's and to the realistic model created were compiled in [29]. To create the slices compiled in this project, Acevedo used the cartography compiled by Rendón, ([51]); the geological plate created by INGEOMINAS 147 scale 1:100 000, ([52]) and the geological map of Aburrá's Valley at scale 1:10 000 compiled by the Microzoning Consortium, ([53]). Also, the slices were also built using information from the following articles: [54], [55] and [56]. The base map compiled is in Fig. B.1 with the different geological formations that appear in the transversal slices. The transversal profiles are shown in Figs. B.2a, B.2b, B.2c, B.2d, B.3a, B.3b, B.3c, B.3c, B.3d, B.4a, B.4b, B.4c, B.4d, B.5a, B.5b, B.5c, B.5d, B.6a, B.6b, B.6c, B.6d and B.7. The slices have the height of the points above the sea level multiplied by 3, so they were transformed accordingly to be used in our algorithms.



GEOLOGIA

- Aluviales
- Depósito de Vertiente
- Sedimentos Terciarios
- Batolito Antioqueño
- Batolito de Ovejas
- Stock de Altavista
- Stock de La Unión
- Stock de San Diego
- Diorita de Heliconia
- Gabros asociados al SFCR
- Metagabros
- Rocas Ultramáficas
- Secuencia Volcánica
- Migmatitas, granulitas y gneises anfibolíticos
- Anfibolitas
- Esquistos cuarzo-sericíticos y cloríticos
- Gneises sintectónicos
- Zona de estudio
- Poblaciones
- Fallas
- Drenajes
- Perfiles

Figure B.1: Base Map of Aburrá's Valley



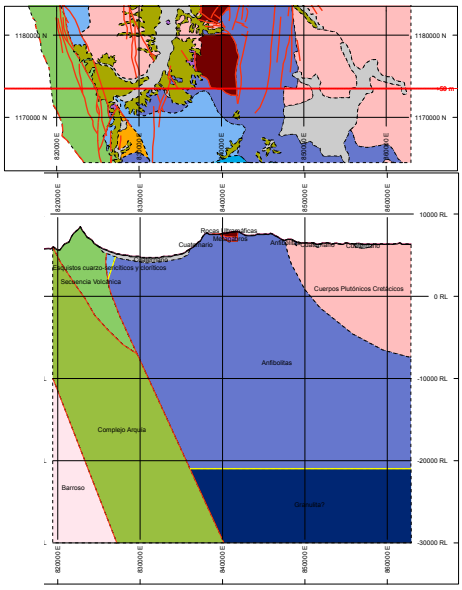
(a) Profile A at coordinate Y = 1165500

(b) Profile B at coordinate Y = 1167500

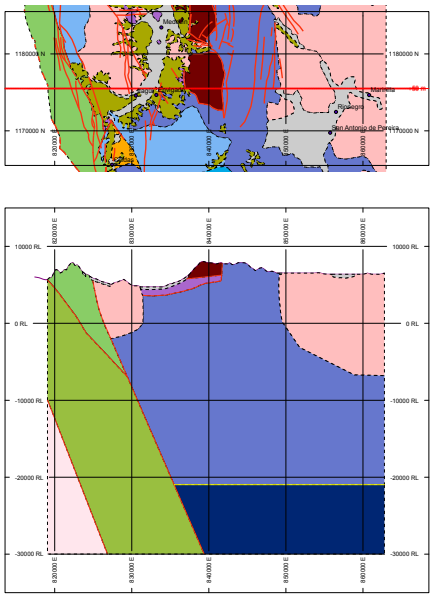
(c) Profile C at coordinate Y = 1169500

(d) Profile D at coordinate Y = 1171500

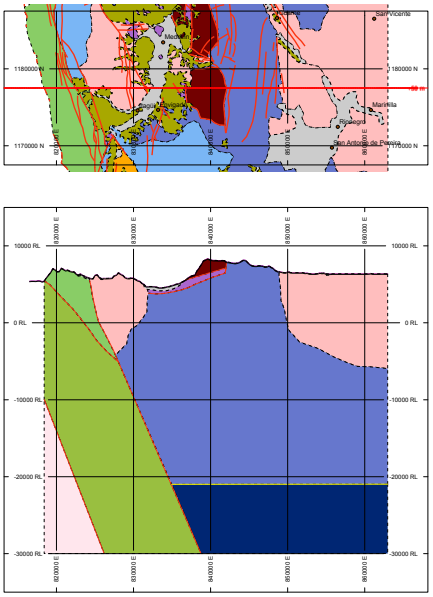
Figure B.2: Profiles A to D



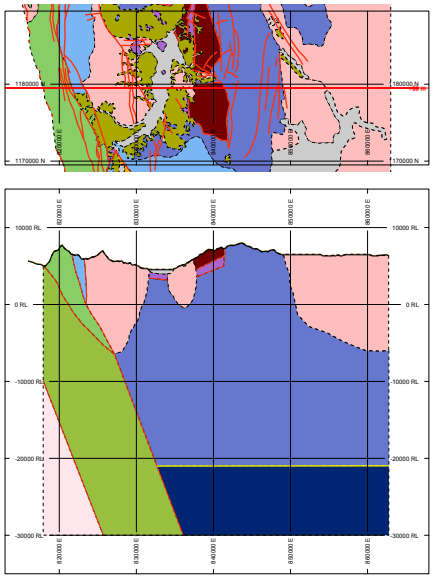
(a) Profile E at coordinate Y = 1173500



(b) Profile F at coordinate Y = 1175500

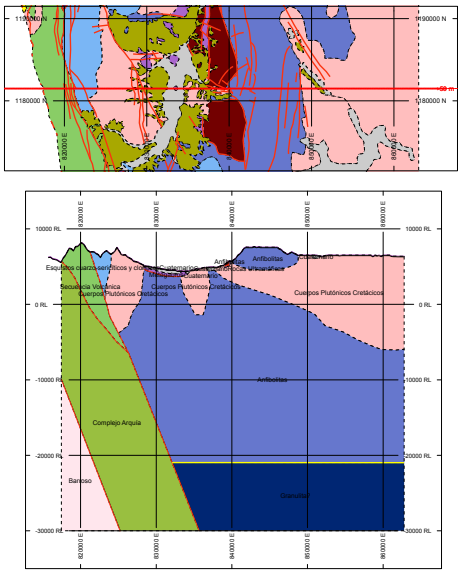


(c) Profile G at coordinate Y = 1177500

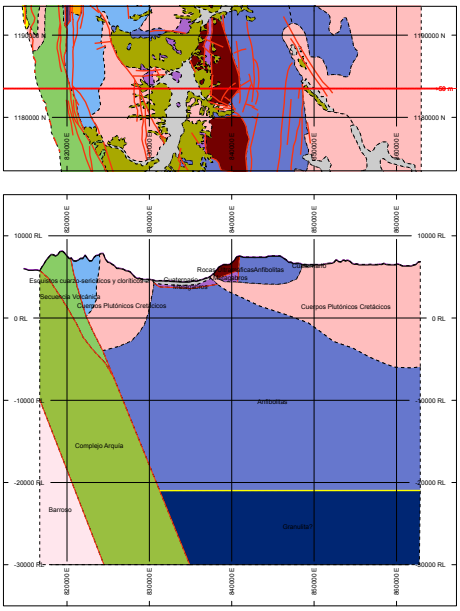


(d) Profile H at coordinate Y = 1179500

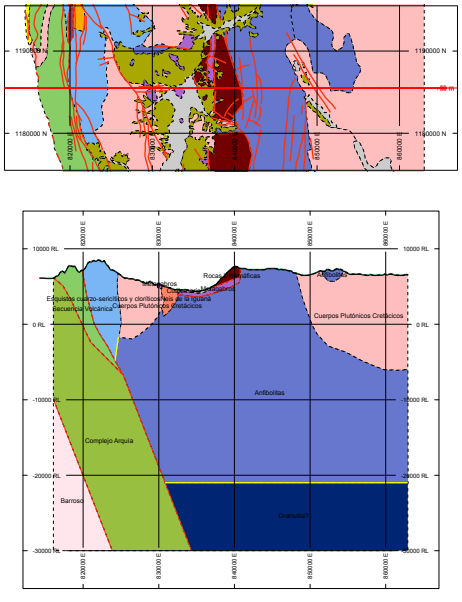
Figure B.3: Profiles E to H



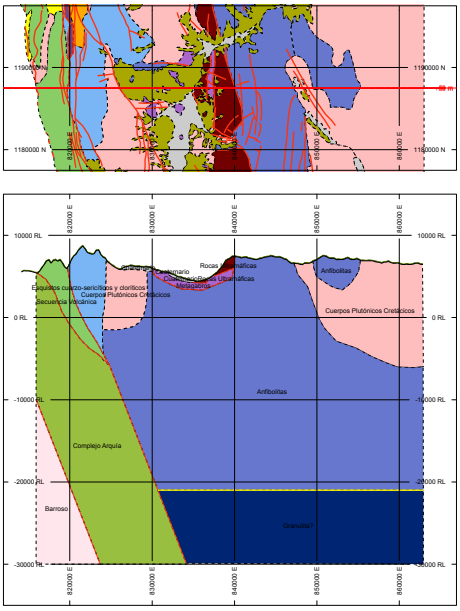
(a) Profile I at coordinate Y = 1181500



(b) Profile J at coordinate Y = 1183500

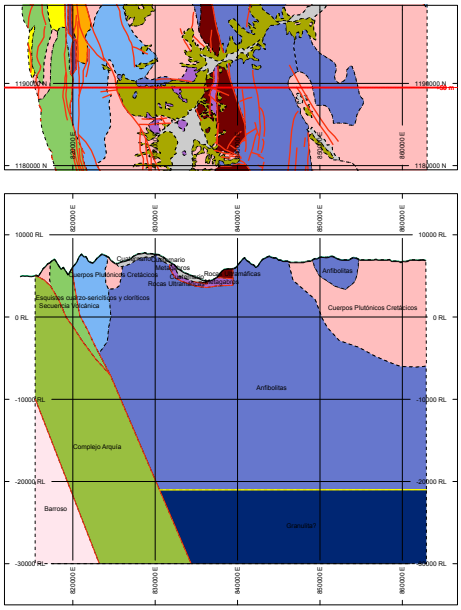


(c) Profile K at coordinate Y = 1185500

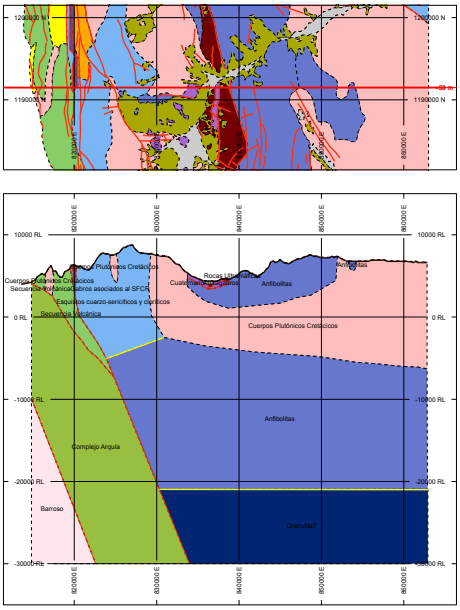


(d) Profile L at coordinate Y = 1187500

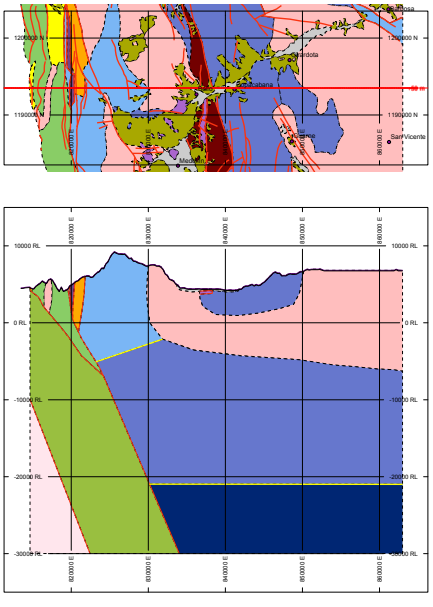
Figure B.4: Profiles I to L



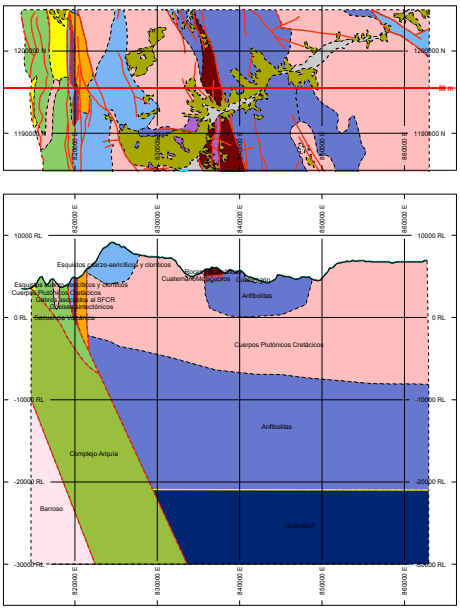
(a) Profile M at coordinate Y = 1189500



(b) Profile N at coordinate Y = 1191500

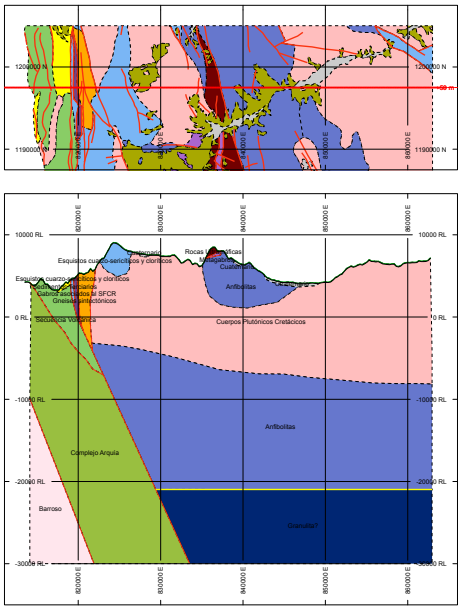


(c) Profile O at coordinate Y = 1193500

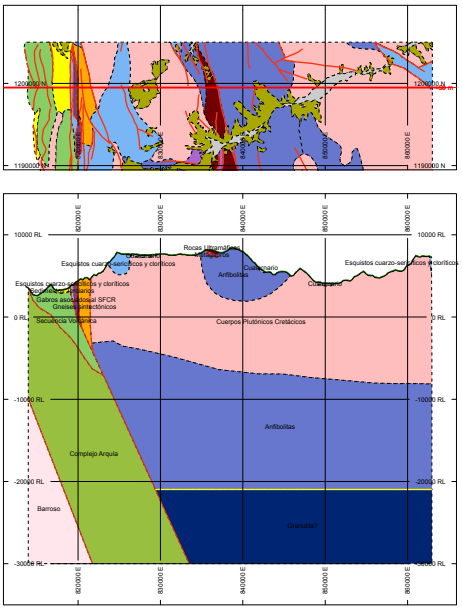


(d) Profile P at coordinate Y = 1195500

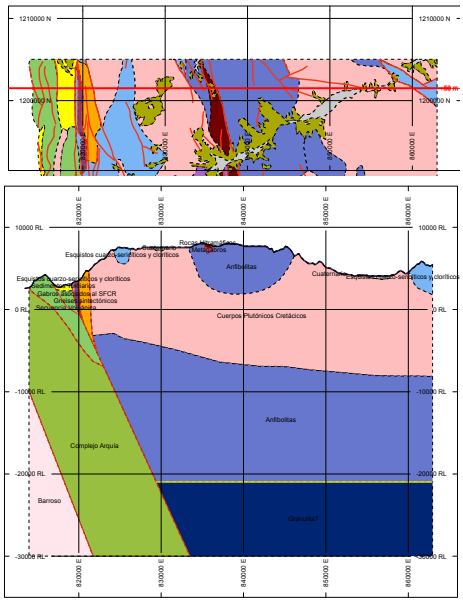
Figure B.5: Profiles M to P



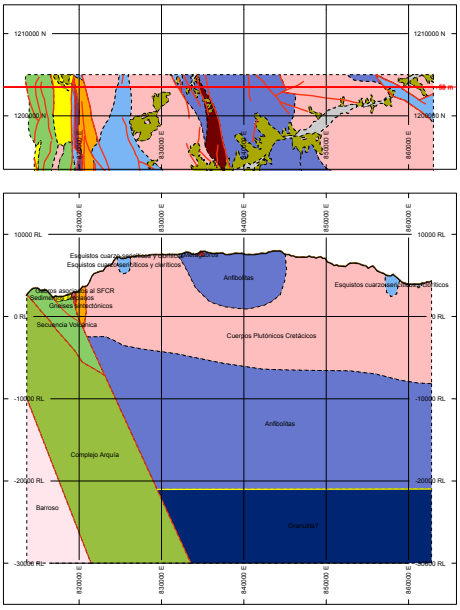
(a) Profile Q at coordinate Y = 1197500



(b) Profile R at coordinate Y = 1199500



(c) Profile S at coordinate Y = 1201500



(d) Profile T at coordinate Y = 1203500

Figure B.6: Profiles Q to T

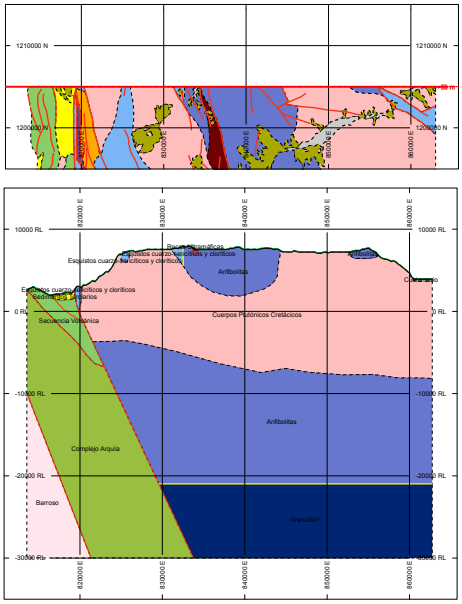


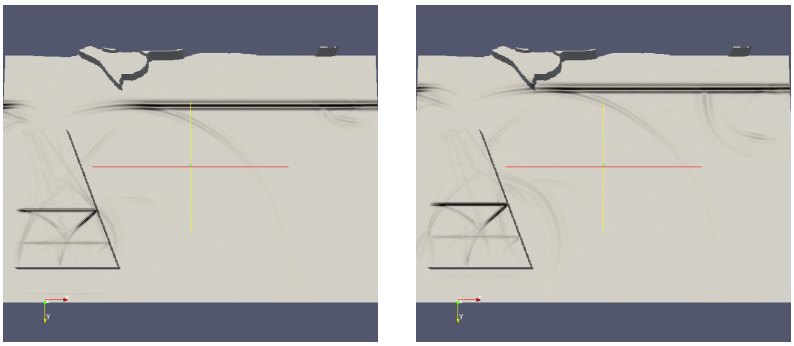
Figure B.7: Profile U at coordinate Y = 1197500

Appendix C

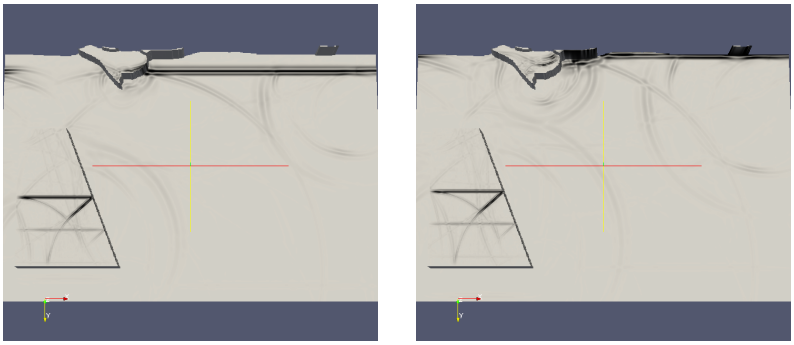
Results at other latitudes of Aburra's Valley

C.1 Cut at Y = 1175000

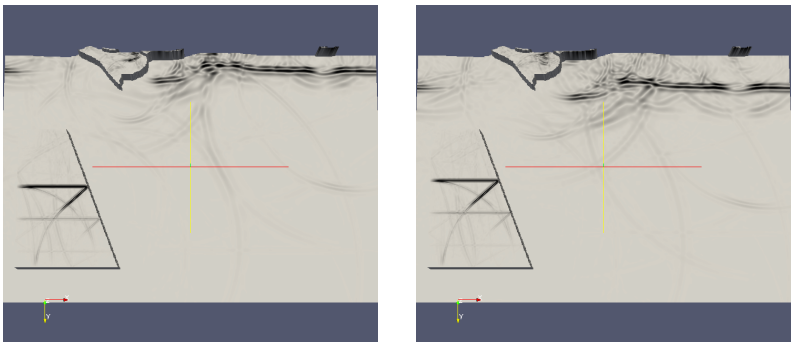
Figure C.1: Slice of Aburra's Valley at Y=1175000.



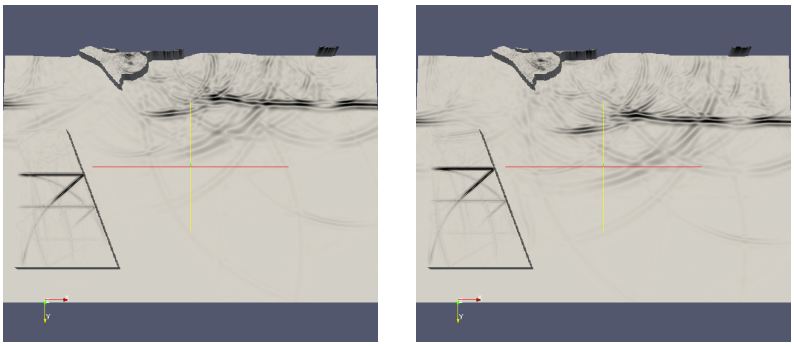
(a) $t = t_1$ (b) $t = t_2$



(c) $t = t_3$ (d) $t = t_4$

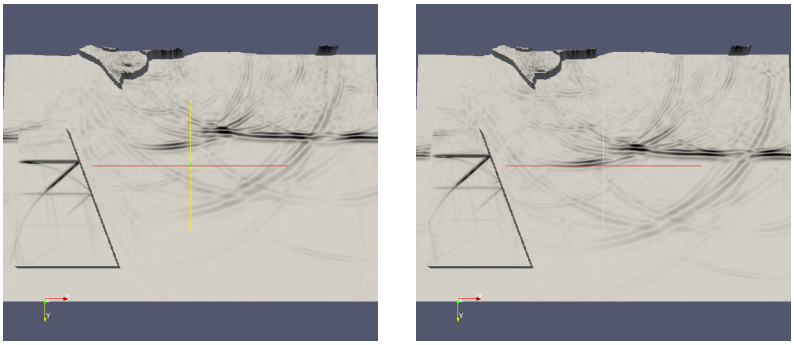


(e) $t = t_5$ (f) $t = t_6$



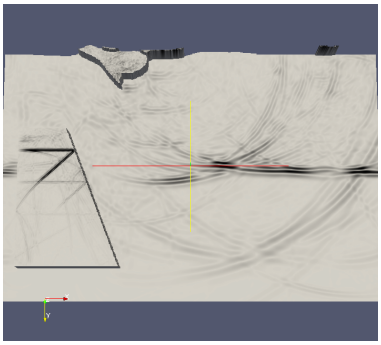
(g) $t = t_7$ (h) $t = t_8$

Figure C.2: Snapshots of an SV-wave incident to Aburraś Valley at 0 degrees, Y coordinate = 1175000. t_1 to t_8

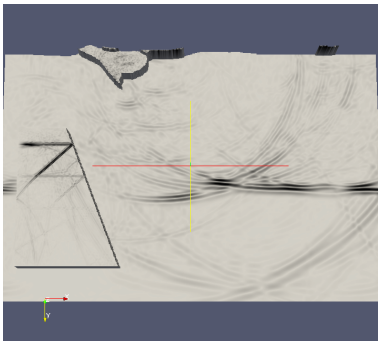


(a) $t = t_9$

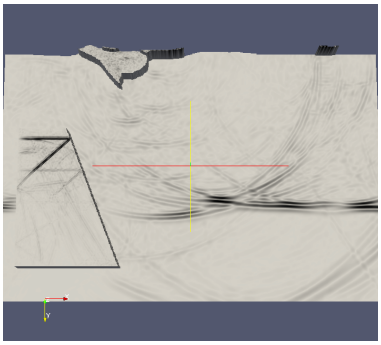
(b) $t = t_{10}$



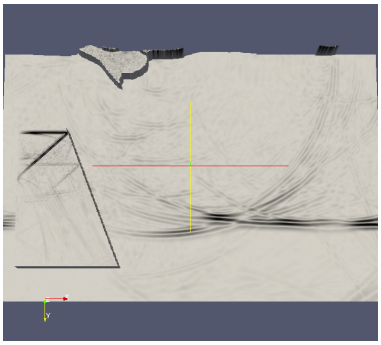
(c) $t = t_{11}$



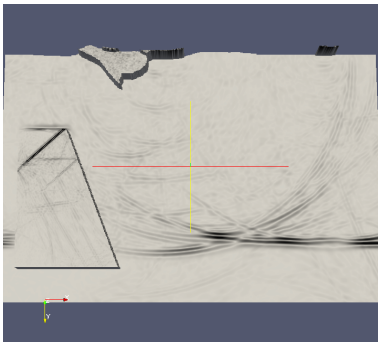
(d) $t = t_{12}$



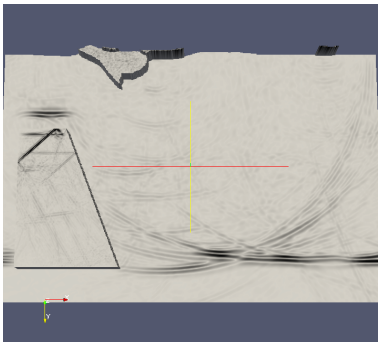
(e) $t = t_{13}$



(f) $t = t_{14}$



(g) $t = t_{15}$



(h) $t = t_{16}$

Figure C.3: Snapshots of an SV-wave incident to Aburraś Valley at 0 degrees, Y coordinate = 1175000. t_9 to t_{16}

C.2 Cut at Y = 1180000

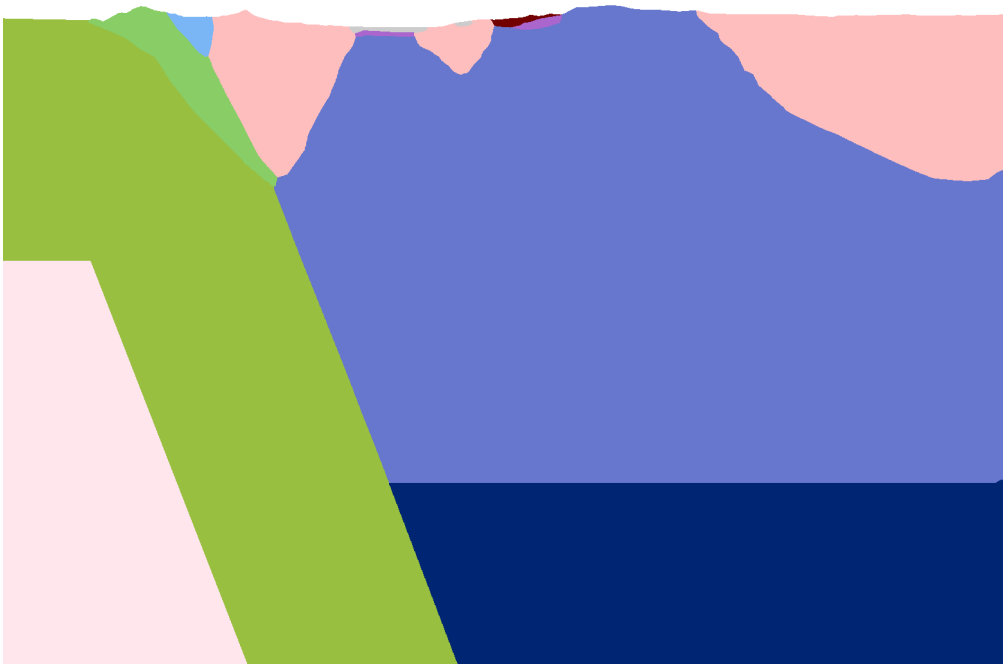
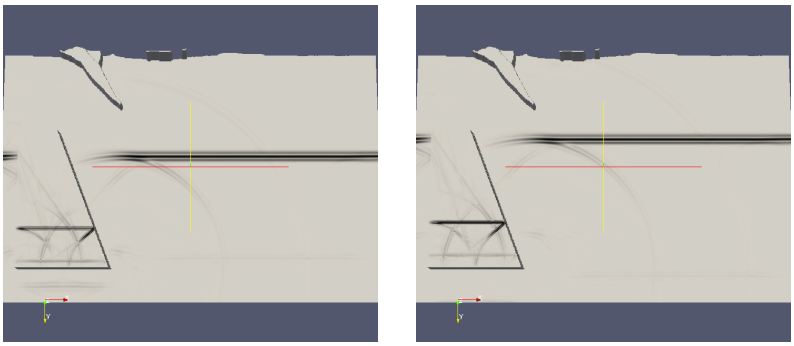
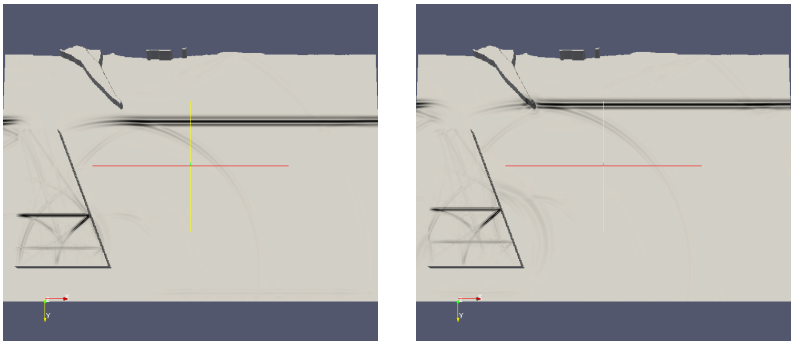


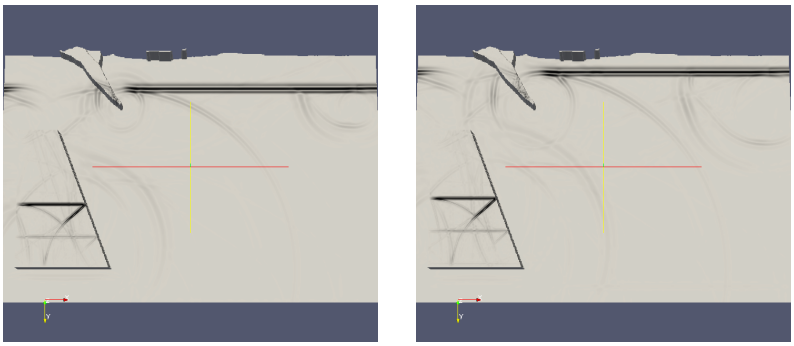
Figure C.4: Slice of Aburra's Valley at Y=1180000.



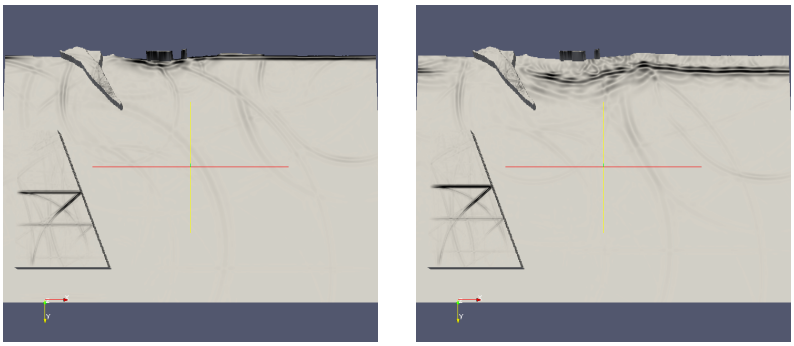
(a) $t = t_1$ (b) $t = t_2$



(c) $t = t_3$ (d) $t = t_4$

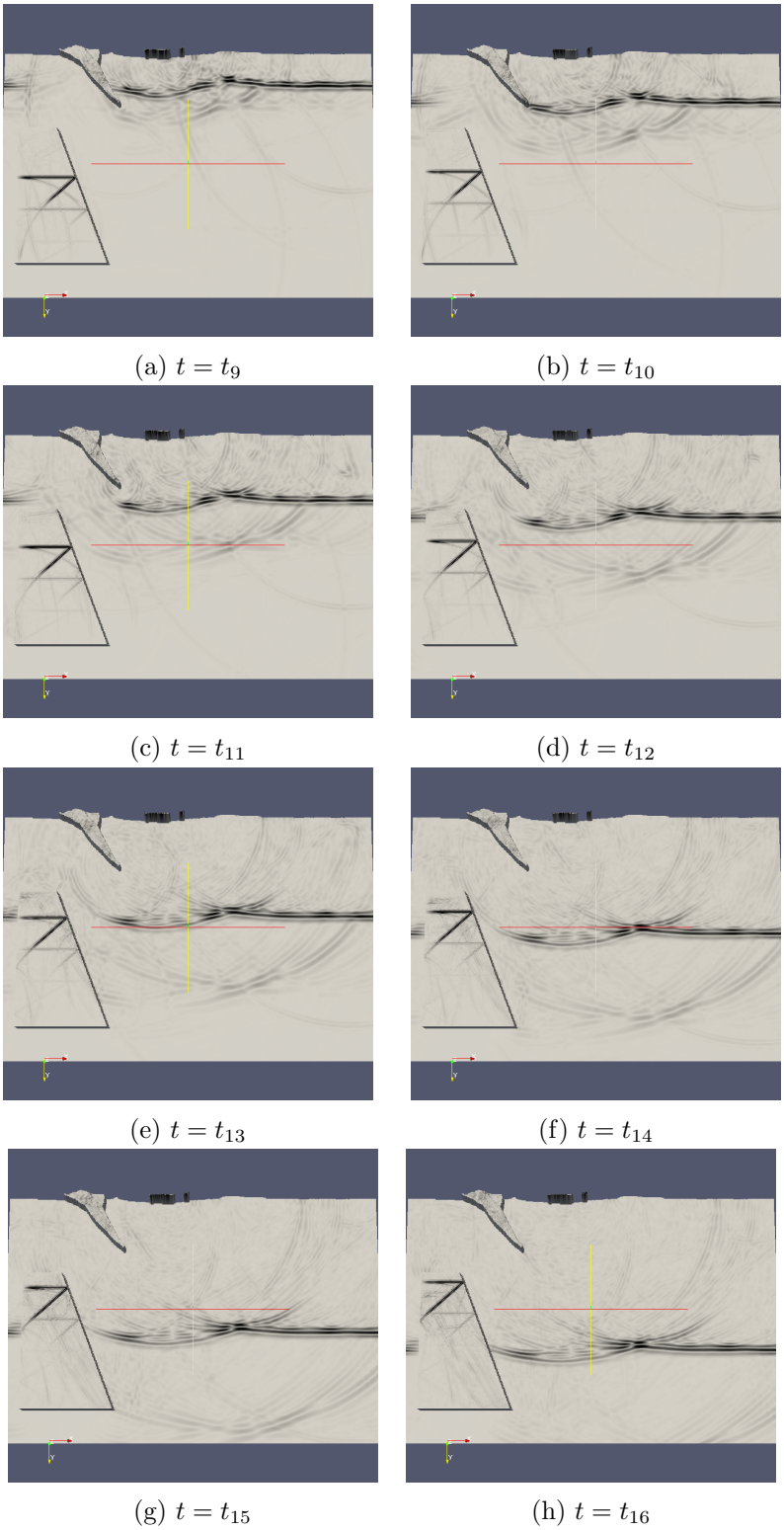


(e) $t = t_5$ (f) $t = t_6$



(g) $t = t_7$ (h) $t = t_8$

Figure C.5: Snapshots of an SV-wave incident to Aburraś Valley at 0 degrees, Y coordinate = 1180000. t_1 to t_8



(a) $t = t_9$ (b) $t = t_{10}$

(c) $t = t_{11}$ (d) $t = t_{12}$

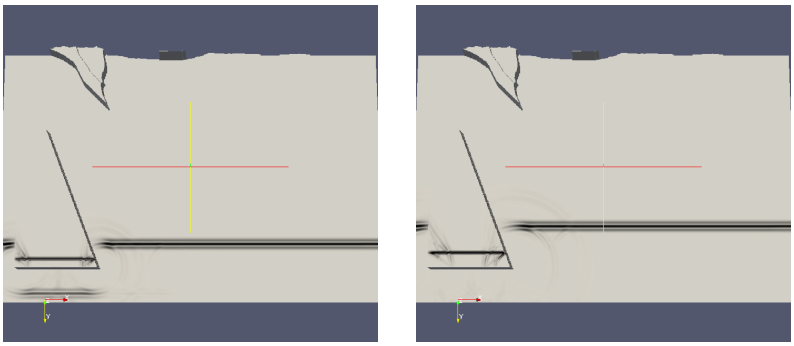
(e) $t = t_{13}$ (f) $t = t_{14}$

(g) $t = t_{15}$ (h) $t = t_{16}$

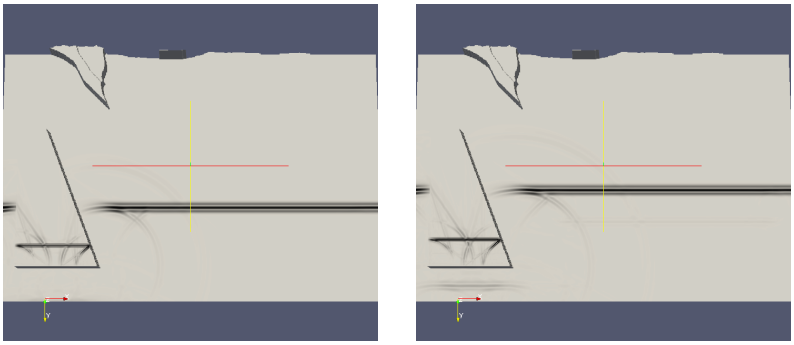
Figure C.6: Snapshots of an SV-wave incident to Aburraś Valley at 0 degrees, Y coordinate = 1180000. t_9 to t_{16}

C.3 Cut at Y = 1185000

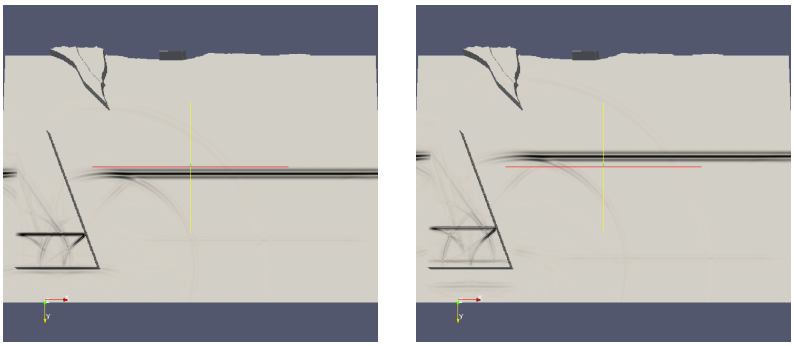
Figure C.7: Slice of Aburra's Valley at Y=1185000.



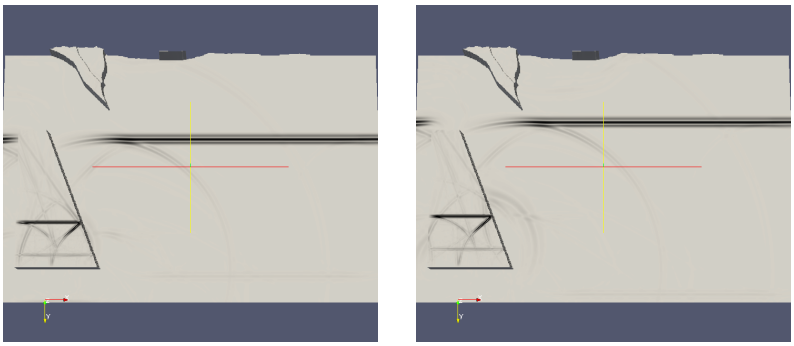
(a) $t = t_1$ (b) $t = t_2$



(c) $t = t_3$ (d) $t = t_4$

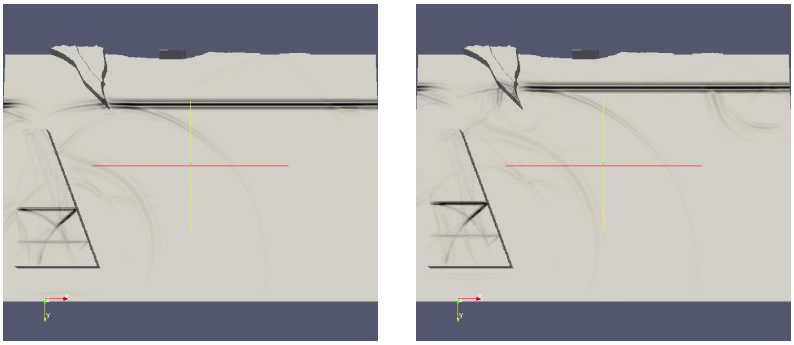


(e) $t = t_5$ (f) $t = t_6$

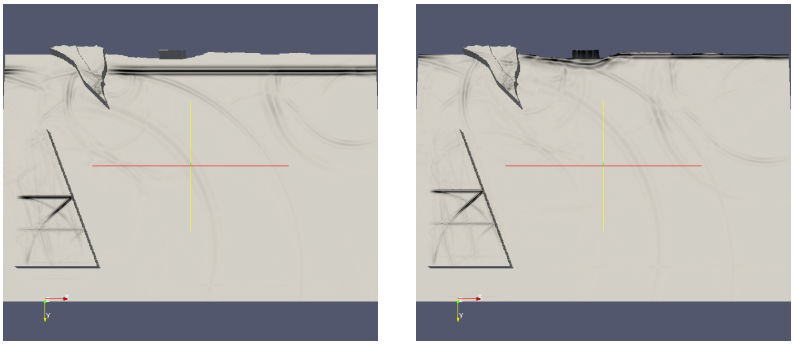


(g) $t = t_7$ (h) $t = t_8$

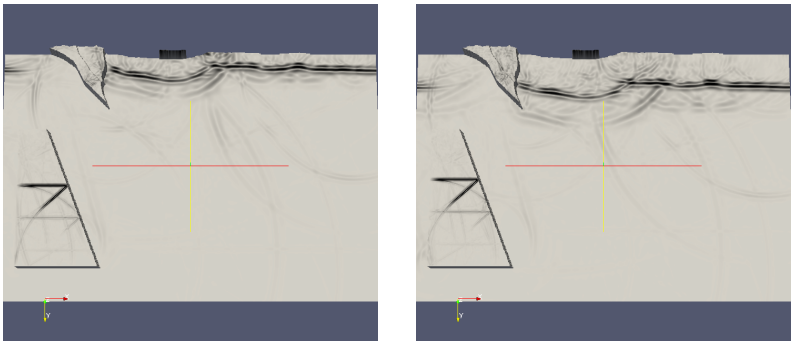
Figure C.8: Snapshots of an SV-wave incident to Aburraś Valley at 0 degrees, Y coordinate = 1185000. t_1 to t_8



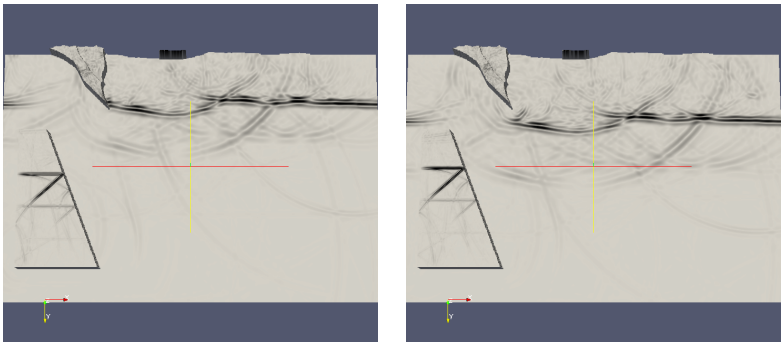
(a) $t = t_9$ (b) $t = t_{10}$



(c) $t = t_{11}$ (d) $t = t_{12}$



(e) $t = t_{13}$ (f) $t = t_{14}$

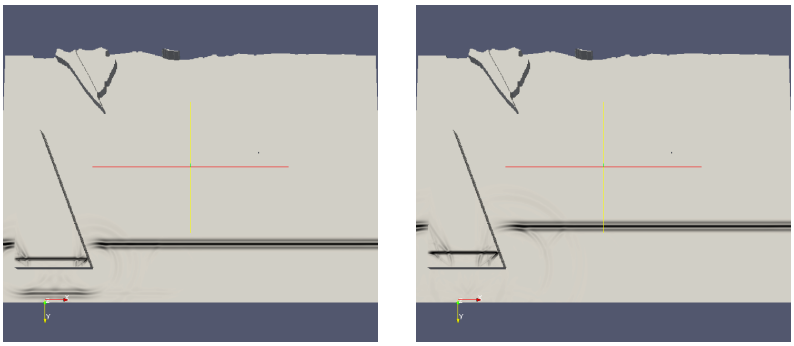


(g) $t = t_{15}$ (h) $t = t_{16}$

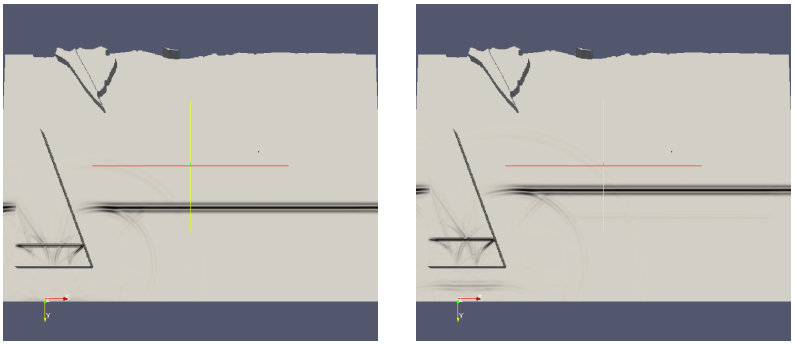
Figure C.9: Snapshots of an SV-wave incident to Aburraś Valley at 0 degrees, Y coordinate = 1185000. t_9 to t_{16}

C.4 Cut at Y = 1190000

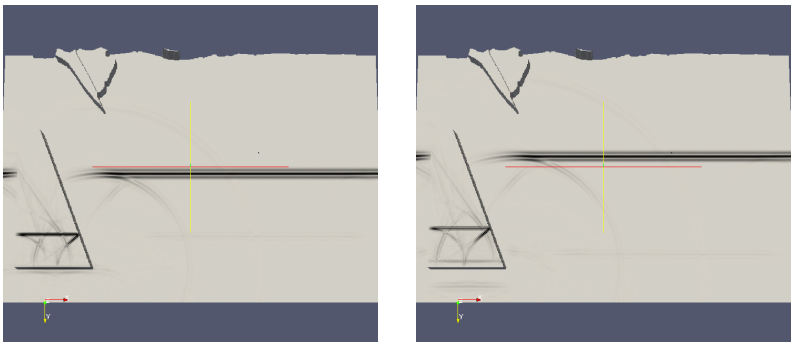
Figure C.10: Slice of Aburra's Valley at Y=1190000.



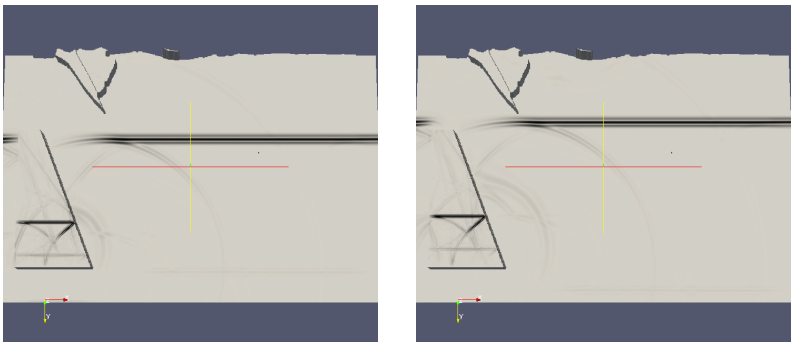
(a) $t = t_1$ (b) $t = t_2$



(c) $t = t_3$ (d) $t = t_4$

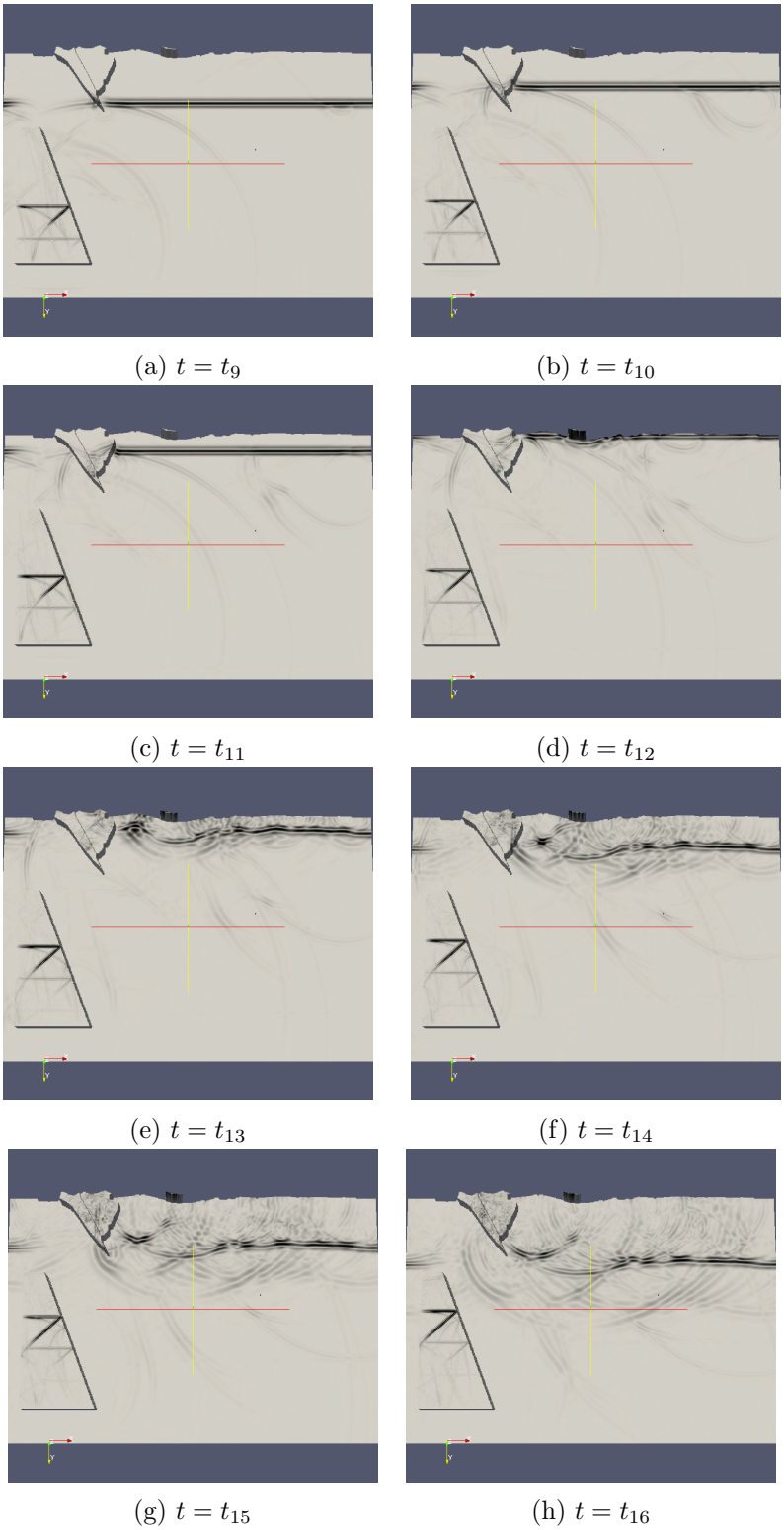


(e) $t = t_5$ (f) $t = t_6$



(g) $t = t_7$ (h) $t = t_8$

Figure C.11: Snapshots of an SV-wave incident to Aburraś Valley at 0 degrees, Y coordinate = 1190000. t_1 to t_8



(a) $t = t_9$ (b) $t = t_{10}$

(c) $t = t_{11}$ (d) $t = t_{12}$

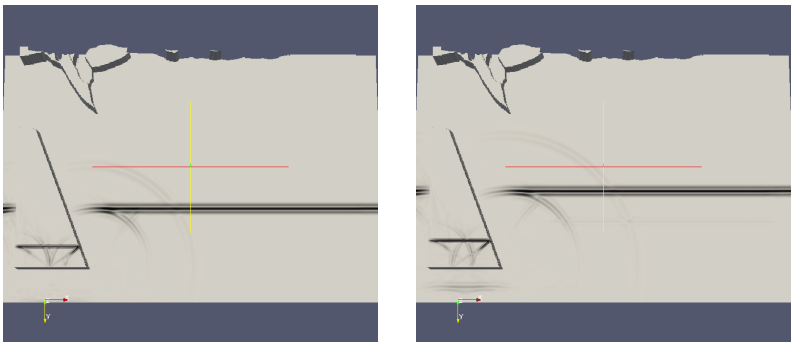
(e) $t = t_{13}$ (f) $t = t_{14}$

(g) $t = t_{15}$ (h) $t = t_{16}$

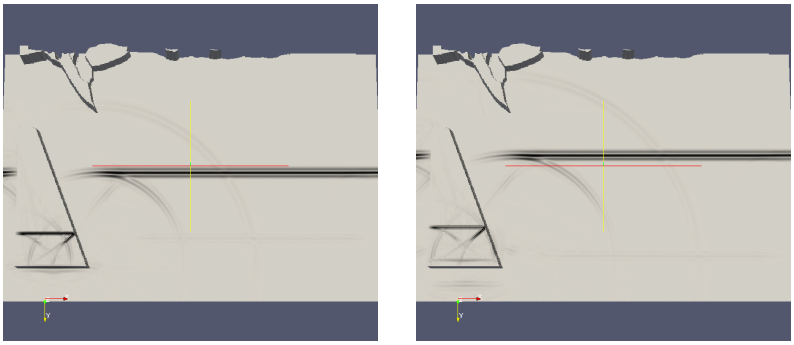
Figure C.12: Snapshots of an SV-wave incident to Aburraś Valley at 0 degrees, Y coordinate = 1190000. t_9 to t_{16}

C.5 Cut at Y = 11950000

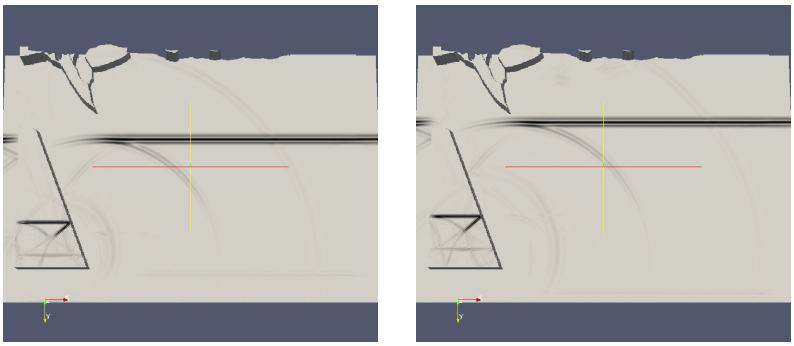
Figure C.13: Slice of Aburra's Valley at Y=1195000.



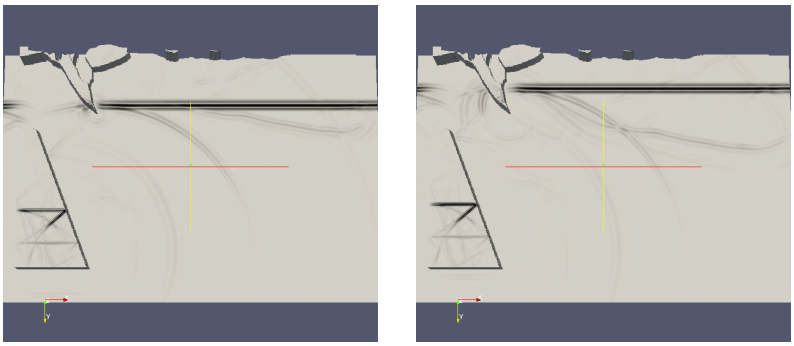
(a) $t = t_1$ (b) $t = t_2$



(c) $t = t_3$ (d) $t = t_4$

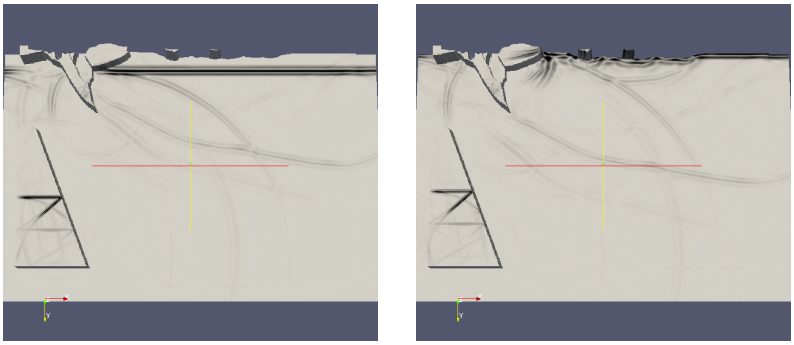


(e) $t = t_5$ (f) $t = t_6$



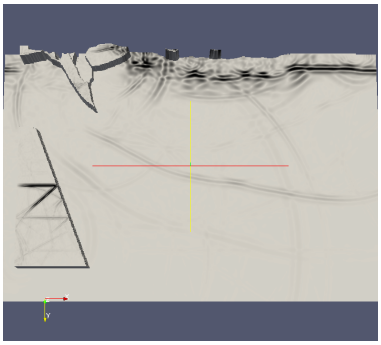
(g) $t = t_7$ (h) $t = t_8$

Figure C.14: Snapshots of an SV-wave incident to Aburraś Valley at 0 degrees, Y coordinate = 1195000. t_1 to t_8

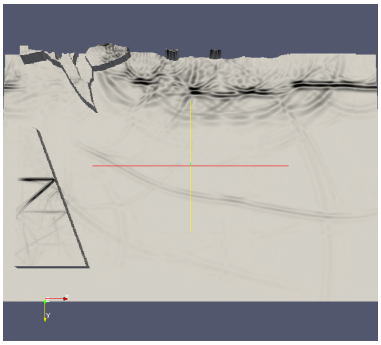


(a) $t = t_9$

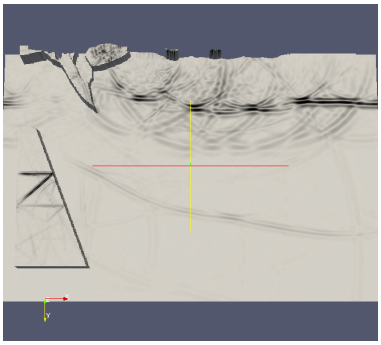
(b) $t = t_{10}$



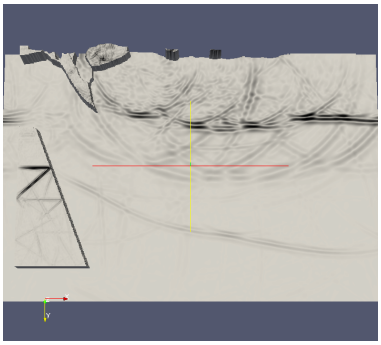
(c) $t = t_{11}$



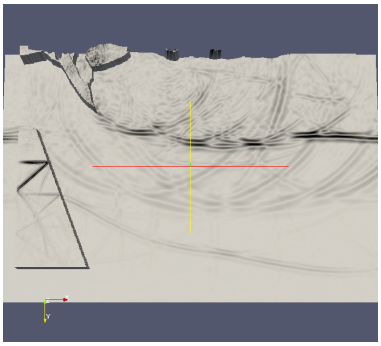
(d) $t = t_{12}$



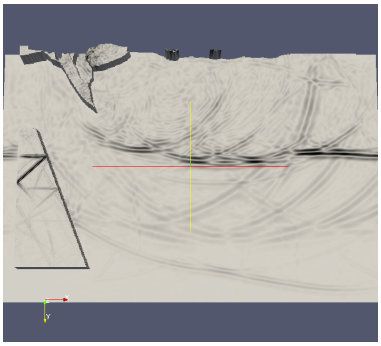
(e) $t = t_{13}$



(f) $t = t_{14}$



(g) $t = t_{15}$



(h) $t = t_{16}$

Figure C.15: Snapshots of an SV-wave incident to Aburraś Valley at 0 degrees, Y coordinate = 1195000. t_9 to t_{16}

Bibliography

[1] P. Moczo, J. Kristek *et al.*, "The Finite difference and finite element modelling of seismic wave propagation and earthquake motion," *Acta Physica Slovaca*, vol. 57, no. 2, pp. 177–406, 2007.

[2] P. Moczo, J. Kristek, and E. Bystrický, "Efficiency and Optimization of the 3-D Finite-Difference Modeling of Seismic Ground Motion," *Journal of Computational Acoustics*, vol. 09, no. 02, pp. 593–609, Jun. 2001.

[3] D. Roten, K. B. Olsen *et al.*, "3D Simulations of M 7 Earthquakes on the Wasatch Fault, Utah, Part I: Long-Period (0-1 Hz) Ground Motion," *Bulletin of the Seismological Society of America*, vol. 101, no. 5, pp. 2045–2063, Sep. 2011.

[4] Y. Cui, R. Moore *et al.*, "Enabling very-large scale earthquake simulations on parallel machines," in *7th International Conference on Computational Science*. Springer-Verlag, 2007, pp. 46–53.

[5] Y. Cui, K. Olsen *et al.*, "Scalable earthquake simulation on petascale supercomputers," *High Performance Computing, Networking, Storage and Analysis (SC)*, no. November, pp. 1–20, 2010.

[6] J. Zhou, Y. Cui *et al.*, "Multi-GPU Implementation of a 3D Finite Difference Time Domain Earthquake Code on Heterogeneous Supercomputers," *Procedia Computer Science*, vol. 18, pp. 1255–1264, Jan. 2013.

[7] D. Komatitsch, S. Tsuboi *et al.*, "Earthquake simulation on the Earth Simulator," in *Proceedings of the 2003 ACM/IEEE conference on Supercomputing*, no. November, 2003, p. 4.

[8] E. Chaljub, D. Komatitsch *et al.*, "Spectral-element analysis in seismology," *Advances in Geophysics*, vol. 48, pp. 365–419, 2007.

[9] D. Komatitsch, D. Michéa, and G. Erlebacher, "Porting a high-order finite-element earthquake modeling application to NVIDIA graphics cards using CUDA," *Journal of Parallel and Distributed Computing*, vol. 69, no. 5, pp. 451–460, May 2009.

[10] D. Komatitsch, Q. Liu *et al.*, "Simulations of ground motion in the Los Angeles basin based upon the spectral-element method," *Bulletin of the Seismological Society of America*, vol. 94, no. 1, pp. 187–206, 2004.

[11] H. Bao, J. Bielak *et al.*, "Earthquake ground motion modeling on parallel computers," in *Proceedings of the 1996 ACM/IEEE conference on Supercomputing (CDROM) - Supercomputing '96*, ACM, Ed. New York, New York, USA: ACM Press, 1996, p. 13.

[12] S. E. Benzley, E. Perry *et al.*, "A comparison of all hexagonal and all tetrahedral finite element meshes for elastic and elasto-plastic analysis," in *Proceedings, 4th International Meshing Roundtable*, 1995, pp. 179—-191.

[13] E. J. Kim, J. Bielak, and O. Ghattas, "Large-scale northridge earthquake simluation using octree-based multiresolution mesh method," in *Proceedings of the 16th ASCE Engineering Mechanics Conference*, ASCE, Ed., vol. 16, Seattle, Washington, 2003, pp. 1–6.

[14] J. Bielak, O. Ghattas, and E. J. Kim, "Parallel octree-based finite element method for large-scale earthquake ground motion simulation," *Computer Modeling in Engineering and sciences*, vol. 10, no. 2, pp. 99–112, 2005.

[15] R. Taborda, J. López *et al.*, "Speeding up finite element wave propagation for large-scale earthquake simulations," *Parallel Data Laboratory Technical Report*, 2010.

[16] J. Bielak, V. Akcelik *et al.*, "High resolution forward and inverse earthquake modeling on terascale computers," in *Supercomputing, 2003 ACM/IEEE Conference*, ACM/IEEE, Ed., no. March 2005, Phoenix, Arizona, 2003, pp. 52–52.

[17] R. Taborda and J. Bielak, "Ground-Motion Simulation and Validation of the 2008 Chino Hills, California, Earthquake," *Bulletin of the Seismological Society of America*, vol. 103, no. 1, pp. 131–156, Feb. 2013.

[18] D. L. Restrepo, "Effect of Topography on 3D Seismic Ground Motion Simulation with an Application to the Valley of Aburra in Antioquia , Colombia," 2013.

[19] J. Lysmer and G. Kuhlemeyer, "Finite Dynamic Model For Infinite Media," *Journal of the Engineering Mechanics Division*, vol. 95, no. 4, pp. 859–878, 1969.

[20] D. P. Young, R. G. Melvin, and M. B. Bieterman, "A Locally Refined Rectangular Grid Finite Element Method: Application to Computational Fluid Dynamics and Computational Physics," *Journal of Computational Physics*, vol. 92, pp. 1–66, 1991.

[21] J. Bielak, K. Loukakis *et al.*, "Domain Reduction Method for Three-Dimensional Earthquake Modeling in Localized Regions, Part I : Theory," *Bulletin of the Seismological Society of America*, vol. 93, no. 2, pp. 817–824, 2003.

[22] H. Ryan, "A Choice of wavelets," Hi-Res Geoconsulting, Tech. Rep., 1994.

[23] H. Kawase, "Time-domain response of a semi-circular canyon for incident SV, P, and Rayleigh waves calculated by the discrete wavenumber boundary element method," *Bulletin of the Seismological Society of America*, vol. 78, no. 4, pp. 1415—-1437, 1988.

[24] G. Karypis, "METIS and ParMETIS," in *Encyclopedia of Parallel Computing*, 2011, pp. 1117–1124.

[25] A. Plesch, C. Tape *et al.*, "User Guide for the Southern California Earthquake Center Community Velocity Model : SCEC CVM-H 11 . 9 . 0," Southern California Earthquake Center, Tech. Rep., 2011.

[26] M. P. Süss and J. H. Shaw, "P wave seismic velocity structure derived from sonic logs and industry reflection data in the Los Angeles basin, California." *Journal of Geophysical Research: Solid Earth (19782012)*, vol. 108, no. B3, 2003.

[27] C. Tape, Q. Liu *et al.*, "Adjoint tomography of the southern California crust." *Science (New York, N.Y.)*, vol. 325, no. 5943, pp. 988–92, Aug. 2009.

[28] C. Tape, a. Plesch *et al.*, "Estimating a Continuous Moho Surface for the California Unified Velocity Model," *Seismological Research Letters*, vol. 83, no. 4, pp. 728–735, Jul. 2012.

[29] A. Acevedo Gomez, D. A. Rendón *et al.*, "Construcción de un modelo interpretativo 3D hasta 30 Km de profundidad de la corteza alrededor del Valle de Aburrá," BSc Thesis, EAFIT University, 2012.

[30] J. Udupa and G. Herman, *3D imaging in medicine*, Jan. 1990, vol. 14, no. 1.

[31] G. T. Herman and J. K. Udupa, "Display of 3-D Digital Images: Computational Foundations and Medical Applications," *Computer Graphics and Applications, IEEE*, vol. 3, no. 5, pp. 34—-46, 1983.

[32] E. Artzy, G. Frieder, and G. T. Herman, "The Theory, Design, Implementation and Evaluation of a Three-Dimensional Surface Detection Algorithm," *Computer graphics and image processing*, vol. 14, no. 3, pp. 2—-9, 1981.

[33] J. Wilhelms and A. V. Gelder, "Topological considerations in isosurface generation extended abstract," in *VVS '90 Proceedings of the 1990 workshop on Volume visualization*, vol. 24, no. 5. ACM New York, NY, USA, 1990, pp. 79–86.

[34] S. Y. Chen, W. C. Lin *et al.*, "Improvement on dynamic elastic interpolation technique for reconstructing 3-D objects from serial cross sections [biomedical application]." *IEEE transactions on medical imaging*, vol. 9, no. 1, pp. 71–83, Jan. 1990.

[35] D. R. Ney, E. K. Fishman *et al.*, "Volumetric Rendering of Computed Tomography Data : Principles and Techniques," *Computer Graphics and Applications, IEEE*, vol. 10, no. 2, pp. 24—-32, 1990.

[36] H. Wackernagel, *Multivariate geostatistics*, 2003.

[37] M. Stytz and R. Parrott, "Using Kriging for 3D Medical Imaging," *Computerized Medical Imaging and Graphics*, vol. 17, no. 6, pp. 421–442, 1993.

[38] S. P. Raya and J. K. Udupa, "Shape-based interpolation of multidimensional objects." *IEEE transactions on medical imaging*, vol. 9, no. 1, pp. 32–42, Jan. 1990.

[39] G. Herman, J. Zheng, and C. Bucholtz, "Shape-Based Interpolation," *IEEE Computer Graphics and . . .*, vol. 12, no. 3, pp. 69—-79, 1992.

[40] W. Lorensen and H. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *ACM Siggraph Computer Graphics*, 1987.

[41] T. Lewiner, H. Lopes *et al.*, "Efficient Implementation of Marching Cubes' Cases with Topological Guarantees," *Journal of Graphics Tools*, vol. 8, no. 2, pp. 1–15, Jan. 2003.

[42] T. S. Newman and H. Yi, "A survey of the marching cubes algorithm," *Computers & Graphics*, vol. 30, no. 5, pp. 854–879, Oct. 2006.

[43] G. Barequet and M. Sharir, "Piecewise-Linear Interpolation between Polygonal Slices," *Computer vision and image understanding*, vol. 63, no. 2, pp. 251–272, 1996.

[44] C. L. Bajaj, E. J. Coyle, and K.-N. Lin, "Arbitrary topology shape reconstruction from planar cross sections," *Graphical models and image processing*, vol. 58, no. 6, pp. 534—-543, 1996.

[45] O. Ruiz, C. Cadavid, and M. Granados, "2D shape similarity as a complement for VoronoiDelone methods in shape reconstruction," *Computers & Graphics*, vol. 29, no. 1, pp. 81–94, 2005.

[46] H. Edelsbrunner and N. R. Shah, "Triangulating topological spaces," in *Proceedings of the 10th annual symposium on Computational Geometry - SCG '94*. New York, New York, USA: ACM Press, 1994, pp. 285–292.

[47] H. Edelsbrunner and E. P. Mücke, "Three-dimensional Alpha Shapes," *ACM Transactions on Graphics (TOG)*, vol. 13, no. 1, pp. 43–72, 1994.

[48] H. Edelsbrunner and J. Harer, *Computational topology: an introduction*, North Carolina, 2010.

[49] G. Zachmann and E. Langetepe, *Geometric data structures for computer graphics*, ACM, Ed. Bonn: ACM New York, NY, USA, 2003.

[50] J. Bielak and P. Christiano, "On the effective seismic input for nonlinear soilstructure interaction systems," *Earthquake engineering & structural dynamics*, vol. 12, no. March 1982, pp. 107–119, 1984.

[51] D. A. Rendón, "Tectonic and sedimentary evolution of the Upper Aburrá Valley, northern Colombian Andes," Ph.D. dissertation, University of Shimane, Japan, 2003.

[52] INGEOMINAS, "Mapa Geológico Plancha 147 Medellín Oriental, escala 1:50.000, memoria explicativa," INGEOMINAS, Medellín, Tech. Rep., 2005.

[53] Consorcio de Microzonificación, "Microzonificación y evaluación del riesgo sísmico del Valle de Aburrá. Informe Final," Consorcio de Microzonificación, Medellín, Tech. Rep., 2006.

[54] A. Chica, J. A. Buitrago *et al.*, "Codificación Sismológica de un Segmento de la Falla Cauca-Almaguer y sus Aplicaciones en el Departamento de Antioquia," *Revista Académia Colombiana de Ciencias*, vol. 28, no. 102, pp. 53–70, 2003.

[55] C. A. Vargas and E. al, "New Geological and Geophysical Contributions in the Section Ibagué-Armenia, Cordillera Central, Colombia," *Earth Sciences Research Journal*, vol. 9, no. 2, pp. 99–109, 2005.

[56] G. Monsalve, C. Villaraga, and J. E. Idárraga, "Inferencias acerca de la estructura sísmica de la litosfera superior bajo el Valle de Aburrá usando registros de redes acelerográficas," *Boletín de Ciencias de la Tierra*, vol. 28, pp. 77–94, 2010.