WAVES: Explicit Parallelized Finite Element Solver for Wave Propagation Analysis

Juan Gomez and Juan Carlos Vergara

November 8, 2017

1 Introduction

Topographic effects have been shown to play a major role in the determination of the seismic ground motions at a site. In the idealized case of homogeneous media submitted to horizontally polarized shear waves there is a relatively large family of analytic or semi-analytic frequency domain solutions. However in the more general case of in-plane waves or in actual three-dimensional topographies the problem needs to be solved numerically. In particular, algorithms based upon the finite element method (FEM) have the advantage of allowing the incorporation of arbitrary boundary conditions and material models. This article describes the explicit finite element code **WAVES** which has been created as part of the research program on topographic effects conducted at the research lab Grupo de Mecanica Aplicada at Universidad EAFIT. Although the code has been originally implemented for the solution of plane wave scattering problems in the time domain, it is also possible to conduct analysis of generalized dynamic problems in 2D and 3D domains.

In simple terms the computer program finds the displacement time history for arbitrary two-dimensional and three-dimensional domains discretized into finite elements. As its main features the code allows to conduct plane wave analysis using as input excitation the domain reduction (DRM) technique formulated by **?**, while the time discretization corresponds to a central difference scheme (having diagonal mass matrices) with uncoupled equations for each degree of freedom.

In the first part of the report we focus on the theoretical aspects of the DRM approach, including the formulation of the elastodynamics scattering problem. This theoretical discussion is presented directly in matrix form assuming that the reader is experienced with finite element algorithms. Following the discussion of the DRM method we present the time-domain discretization resulting in an explicit solution scheme. In part 4 of the report we describe the most relevant structural aspects of the program with particular emphasis on the addition of user defined finite elements and material models. In the final part of the report we show the model creation process using as study problem the case of a rectangular-shaped canyon embedded in an elastic half-space. Since the model involves a large number of elements the required input files are created with the aid of third party software.

2 Formulation of the plane wave scattering problem

Classical formulation of the scattering problem

In order to have a general context of the physical basis of the DRM technique formulated by **?** it is convenient to describe the so-called scattering problem in elastodynamics. We will use an integral equation approach based on the elastodynamics representation theorem (**?**). The physical problem is schematized by the top part of fig. 1 which depicts a generalized half-space with domain Ω^+ supporting a scatterer with domain Ω. Both domains are bounded through the perfectly coupled surface Γ and the remote boundary of the half-space Γ^+. Notice that the scatterer also comprises a localized, small-scale topographic feature (e.g., a microzone) embedded into a large topographic irregularity. The scattering problem consists in determining the response of the system when it is submitted to the action of an incident wave.

The domain reduction method, originally formulated by **?** and verified in **?** is a two-step algorithm developed for the simulation of earthquake ground motions in seismic scenarios containing strong variations in wavelength. The fundamental principle underlying the method is the classical partition of the field used in the formulation of scattering problems (**???**) written like ;

$$u = u^0 + w \tag{1}$$

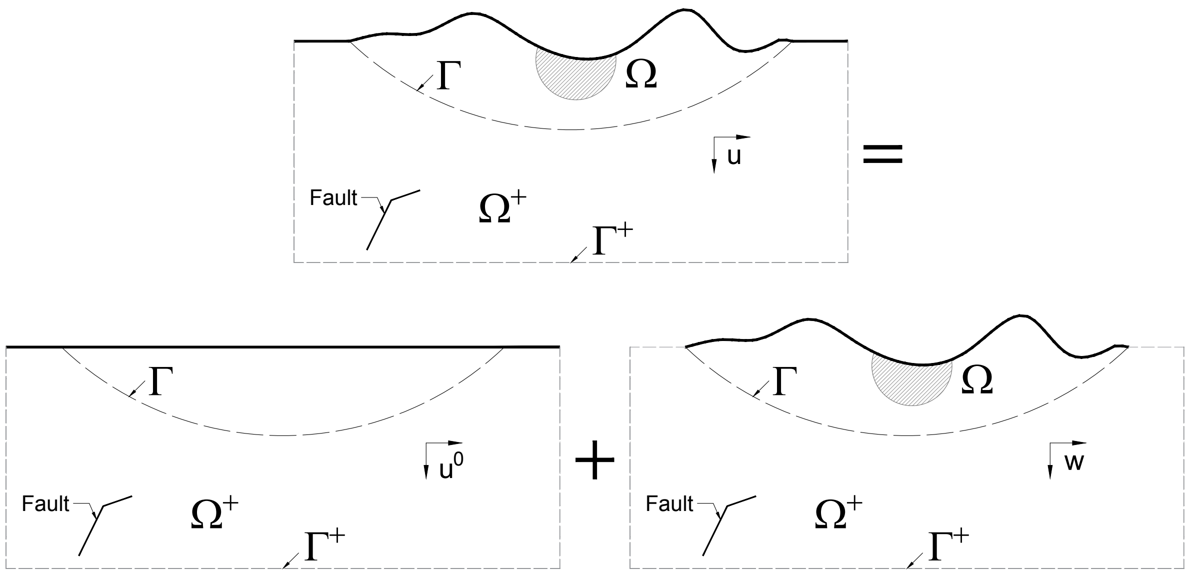


Figure 1: Partition of the field in the classical definition of the scattering problem. The free-field u^0 corresponds to the response of the half-space with the scatterer being removed (bottom left), while the scattered field w would be the additional displacement introduced in the half-space once the scatterer Ω is considered (bottom right).

where u^0 is the free field motion or response of the half-space in the absence of the scatterer and w is a scattered field or relative displacement motion between the total and free-field motion. This free field motion can be obtained in closed-form depending on the nature of the half-space and of the seismic excitation. In the more general case it is also found numerically from the solution of the simpler problem described in the bottom left of fig. 1.

Following the elastodynamics representation theorem the boundary value problem for the total field inside the scatterer is governed by the integral equations;

$$C_{ij}(\vec{\xi})u_j(\vec{\xi}) = \int\limits_{\Gamma} G_{ij}^{FS}(\vec{x};\vec{\xi})t_j(\vec{x})dS(\vec{x}) - \int\limits_{\Gamma^+ + \Gamma} H_{ij}^{FS}(\vec{x};\vec{\xi})u_j(\vec{x})dS(\vec{x}) \quad \text{for } \vec{\xi} \text{ inside } \Gamma^+ \cup \Gamma \tag{2}$$

and where G_{ij}^{FS} and H_{ij}^{FS} are the full-space displacements and tractions Green's tensors respectively while C_{ij} is a tensor which depends on the smoothness of the boundary and u_i and t_i are the total displacements and tractions vectors. Similarly, the relative motion in the half-space w_i is governed by;

$$C_{ij}(\vec{\xi})w_j(\vec{\xi}) = \int\limits_{\Gamma} G_{ij}^{HS}(\vec{x};\vec{\xi})t_j^w(\vec{x})dS(\vec{x}) - \int\limits_{\Gamma} H_{ij}^{HS}(\vec{x};\vec{\xi})w_j(\vec{x})dS(\vec{x}) \quad \text{for } \vec{\xi} \text{ inside } S_F \cup \Gamma^+ \cup \Gamma \tag{3}$$

where now G_{ij}^{HS} and H_{ij}^{HS} are the corresponding Green's tensors for the half-space. Notice that the excitation enters into the problem once the surface compatibility conditions;

$$\begin{aligned} u &= u^0 + w \\ t &+ t^0 + t_w = 0 \end{aligned} \tag{4}$$

are imposed along Γ.

Bielak et al(2003) DRM algorithm

The problem is schematized by the top part of fig. 2 which depicts a generalized half-space with domain Ω^+ supporting a scatterer Ω. Both domains are bounded through the perfectly coupled surface Γ. Notice that the scatterer comprises also a localized, small-scale topographic feature (e.g., a microzone) embedded into a large topographic irregularity. The relevant degrees of freedom have been labeled after **?**. In this work we are interested in conducting SRA at the microzone.

The partitioned equations of motion for the half-space and scatterer (i.e., Ω) read;

$$\begin{bmatrix} M_{ii}^{\Omega} & M_{ib}^{\Omega} \\ M_{bi}^{\Omega} & M_{bb}^{\Omega} \end{bmatrix} \begin{Bmatrix} \ddot{u}_i \\ \ddot{u}_b \end{Bmatrix} + \begin{bmatrix} K_{ii}^{\Omega} & K_{ib}^{\Omega} \\ K_{bi}^{\Omega} & K_{bb}^{\Omega} \end{bmatrix} \begin{Bmatrix} u_i \\ u_b \end{Bmatrix} = \begin{Bmatrix} 0 \\ P_b \end{Bmatrix} \tag{5}$$

$$\begin{bmatrix} M_{bb}^{\Omega^+} & M_{be}^{\Omega^+} \\ M_{eb}^{\Omega^+} & M_{ee}^{\Omega^+} \end{bmatrix} \begin{Bmatrix} \ddot{u}_b \\ \ddot{u}_e \end{Bmatrix} + \begin{bmatrix} K_{bb}^{\Omega^+} & K_{be}^{\Omega^+} \\ K_{eb}^{\Omega^+} & K_{ee}^{\Omega^+} \end{bmatrix} \begin{Bmatrix} u_b \\ u_e \end{Bmatrix} = \begin{Bmatrix} -P_b \\ P_e \end{Bmatrix} \tag{6}$$

where P_b are nodal forces through the coupling surface Γ, P_e represent the loads induced by a seismic source or by an incident plane wave and M and K are finite element mass and stiffness matrices. Coupling eq. (5) and eq. (6) yields the complete system of equations governing the half-space-scatterer system subjected to an exterior seismic source P_e and solved in one step algorithms (i.e., without DRM);

$$
\begin{bmatrix} M_{ii}^{\Omega} & M_{ib}^{\Omega} & 0 \\ M_{bi}^{\Omega} & M_{bb}^{\Omega}+M_{bb}^{\Omega^+} & M_{be}^{\Omega^+} \\ 0 & M_{eb}^{\Omega^+} & M_{ee}^{\Omega^+} \end{bmatrix} \begin{Bmatrix} \ddot{u}_i \\ \ddot{u}_b \\ \ddot{u}_e \end{Bmatrix} + \begin{bmatrix} K_{ii}^{\Omega} & K_{ib}^{\Omega} & 0 \\ K_{bi}^{\Omega} & K_{bb}^{\Omega}+K_{bb}^{\Omega^+} & K_{be}^{\Omega^+} \\ 0 & K_{eb}^{\Omega^+} & K_{ee}^{\Omega^+} \end{bmatrix} \begin{Bmatrix} u_i \\ u_b \\ u_e \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ P_e \end{Bmatrix} \tag{7}
$$

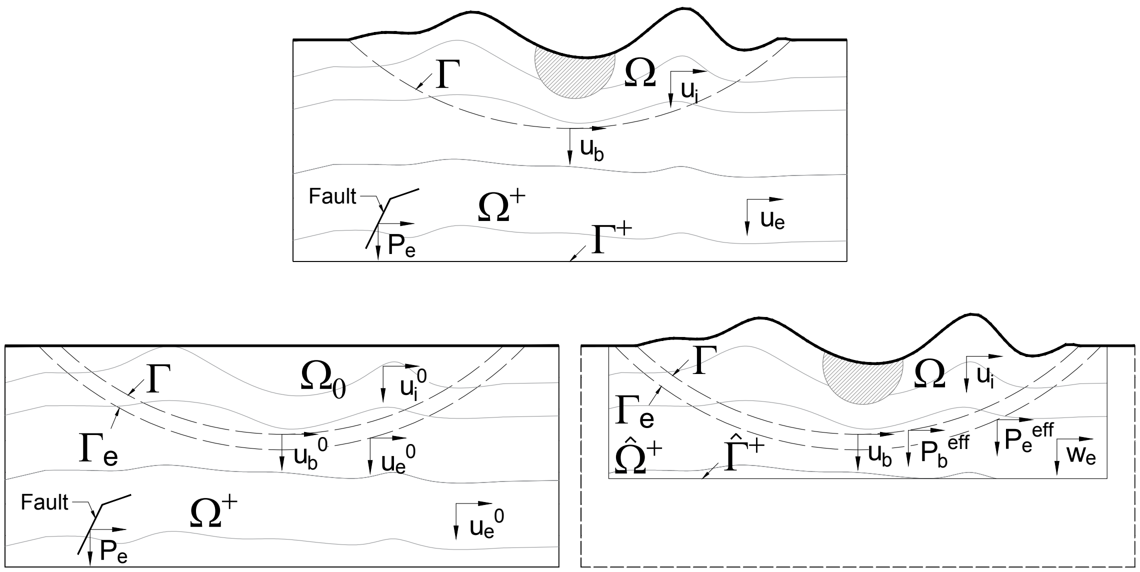


Figure 2: Generalized partition of the total domain and its related fields in the original DRM algorithm. The first analysis step is conducted over the half-space with domain $\Omega^+ \cup \Omega_0$ (bottom left). The resulting motion is then applied as input excitation to the reduced domain $\hat{\Omega}^+ \cup \Omega$ comprising only the localized feature (bottom right).

In the original DRM method the effect of the seismic sources P_e is transferred to the coupling surface Γ using an auxiliary problem (or background structure) constructed after removing from the complete domain the scatterer Ω and replacing it by an arbitrary simplified domain Ω_0 resulting in a generalized half-space $\Omega^+ \cup \Omega_0$ (as shown in the left part of fig. 2). This arbitrary domain is selected in such a way that it is easier to discretize than the original problem. The equations of motion for the domain Ω^+ in the auxiliary problem under the action of the seismic sources P_e yields a generalized free field motion u^0 governed by;

$$
\begin{bmatrix} M_{bb}^{\Omega^+} & M_{be}^{\Omega^+} \\ M_{eb}^{\Omega^+} & M_{ee}^{\Omega^+} \end{bmatrix} \begin{Bmatrix} \ddot{u}_b^0 \\ \ddot{u}_e^0 \end{Bmatrix} + \begin{bmatrix} K_{bb}^{\Omega^+} & K_{be}^{\Omega^+} \\ K_{eb}^{\Omega^+} & K_{ee}^{\Omega^+} \end{bmatrix} \begin{Bmatrix} u_b^0 \\ u_e^0 \end{Bmatrix} = \begin{Bmatrix} -P_b^0 \\ P_e \end{Bmatrix} \tag{8}
$$

allowing to express the seismic sources like;

$$
P_e = M_{eb}^{\Omega^+} \ddot{u}_b^0 + M_{ee}^{\Omega^+} \ddot{u}_e^0 + K_{eb}^{\Omega^+} u_b^0 + K_{ee}^{\Omega^+} u_e^0. \tag{9}
$$

The presence of the terms $M_{ee}^{\Omega^+} \ddot{u}_e^0$ and $K_{ee}^{\Omega^+} u_e^0$ in eq. (9) imply that the free field u^0 must be stored throughout the full domain Ω^+. This inconvenient requirement is dealt with after writing the total field in the exterior part of Ω^+ like a superposition of the free field motion u_e^0 and the relative (or scattered) motion w_e as;

$$u_e = u_e^0 + w_e \tag{10}$$

which yields;

$$\begin{aligned}
&\begin{bmatrix} M_{ii}^{\Omega} & M_{ib}^{\Omega} & 0 \\ M_{bi}^{\Omega} & M_{bb}^{\Omega} + M_{bb}^{\Omega+} & M_{be}^{\Omega+} \\ 0 & M_{eb}^{\Omega+} & M_{ee}^{\Omega+} \end{bmatrix} \begin{Bmatrix} \ddot{u}_i \\ \ddot{u}_b \\ \ddot{w}_e \end{Bmatrix} + \begin{bmatrix} K_{ii}^{\Omega} & K_{ib}^{\Omega} & 0 \\ K_{bi}^{\Omega} & K_{bb}^{\Omega} + K_{bb}^{\Omega+} & K_{be}^{\Omega+} \\ 0 & K_{eb}^{\Omega+} & K_{ee}^{\Omega+} \end{bmatrix} \begin{Bmatrix} u_i \\ u_b \\ w_e \end{Bmatrix} \\
&= \begin{Bmatrix} 0 \\ -M_{be}^{\Omega+} \ddot{u}_e^0 - K_{be}^{\Omega+} u_e^0 \\ P_e - M_{ee}^{\Omega+} \ddot{u}_e^0 - K_{ee}^{\Omega+} u_e^0 \end{Bmatrix}
\end{aligned} \tag{11}$$

After substituting for P_e from eq. (9) into eq. (11) the equations of motion are written in terms of degrees of freedom over a single layer of finite elements in Ω^+ adjacent to Γ. This strip of 1-element width lies between Γ and its adjacent surface Γ_e (see fig. 2);

$$\begin{aligned}
&\begin{bmatrix} M_{ii}^{\Omega} & M_{ib}^{\Omega} & 0 \\ M_{bi}^{\Omega} & M_{bb}^{\Omega} + M_{bb}^{\Omega+} & M_{be}^{\Omega+} \\ 0 & M_{eb}^{\Omega+} & M_{ee}^{\Omega+} \end{bmatrix} \begin{Bmatrix} \ddot{u}_i \\ \ddot{u}_b \\ \ddot{w}_e \end{Bmatrix} + \begin{bmatrix} K_{ii}^{\Omega} & K_{ib}^{\Omega} & 0 \\ K_{bi}^{\Omega} & K_{bb}^{\Omega} + K_{bb}^{\Omega+} & K_{be}^{\Omega+} \\ 0 & K_{eb}^{\Omega+} & K_{ee}^{\Omega+} \end{bmatrix} \begin{Bmatrix} u_i \\ u_b \\ w_e \end{Bmatrix} \\
&= \begin{Bmatrix} 0 \\ -M_{be}^{\Omega+} \ddot{u}_e^0 - K_{be}^{\Omega+} u_e^0 \\ M_{eb}^{\Omega+} \ddot{u}_b^0 + K_{eb}^{\Omega+} u_b^0 \end{Bmatrix}
\end{aligned} \tag{12}$$

The actual domain reduction leading to the method in the formulation from **?** is possible after noticing that all the waves in the exterior region Ω^+ are outgoing. Since the primary interest is in the determination of the response for the local site (i.e., a microzone) this fact suggests that the size of Ω^+ can be drastically reduced. This exterior reduced domain in the DRM algorithm is termed $\hat{\Omega}^+$.

The following points regarding the DRM formulation must be highlighted. First the equations of motion given in eq. (7) when written in terms of the free field motion reads:

$$\begin{aligned}
&\begin{bmatrix} M_{ii}^{\Omega} & M_{ib}^{\Omega} & 0 \\ M_{bi}^{\Omega} & M_{bb}^{\Omega} + M_{bb}^{\Omega+} & M_{be}^{\Omega+} \\ 0 & M_{eb}^{\Omega+} & M_{ee}^{\Omega+} \end{bmatrix} \begin{Bmatrix} \ddot{u}_i \\ \ddot{u}_b \\ \ddot{u}_e \end{Bmatrix} + \begin{bmatrix} K_{ii}^{\Omega} & K_{ib}^{\Omega} & 0 \\ K_{bi}^{\Omega} & K_{bb}^{\Omega} + K_{bb}^{\Omega+} & K_{be}^{\Omega+} \\ 0 & K_{eb}^{\Omega+} & K_{ee}^{\Omega+} \end{bmatrix} \begin{Bmatrix} u_i \\ u_b \\ u_e \end{Bmatrix} \\
&= \begin{Bmatrix} 0 \\ 0 \\ M_{eb}^{\Omega+} \ddot{u}_b^0 + M_{ee}^{\Omega+} \ddot{u}_e^0 + K_{eb}^{\Omega+} u_b^0 + K_{ee}^{\Omega+} u_e^0 \end{Bmatrix}.
\end{aligned} \tag{13}$$

However complete transfer of the seismic sources P_e to the coupling surface Γ is only achieved after one eliminates from eq. (13) the terms $M_{ee}^{\Omega+} \ddot{u}_e^0$ and $K_{ee}^{\Omega+} u_e^0$ which is accomplished by introducing the change of variables resulting after writing the total field in the exterior part of Ω^+ in terms of the free-field and relative motion. For this change of variables to remain valid it is required that the reduced exterior domain $\hat{\Omega}^+$ retains the same material properties as the original exterior domain Ω^+. The resulting DRM approach can be summarized in the following two-step algorithm:

- In step-I the complete seismic domain $\Omega \cup \Omega^+$, comprising the seismic source and microzones of soft material properties is replaced by a simpler domain $\Omega_0 \cup \Omega^+$ which results after removing all the surface topography and localized features.

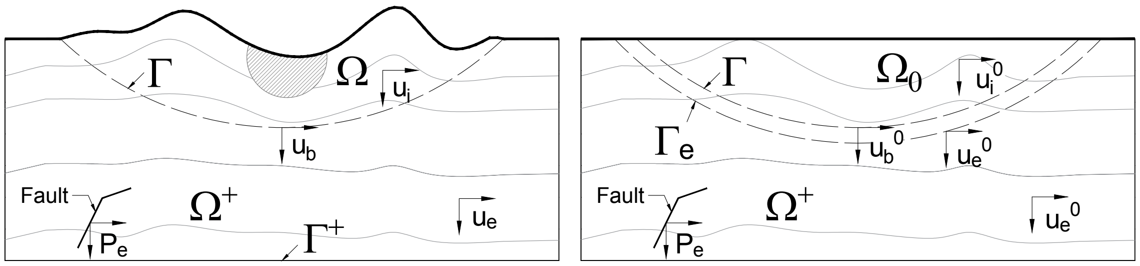


Figure 3: In step I of the DRM algorithm the original domain $\Omega \cup \Omega^+$ (left) is replaced by a simpler domain $\Omega_0 \cup \Omega^+$ (right).

This simpler domain is analyzed, under the action of seismic sources P_e, in order to determine the free field response leading to effective loads located over a boundary adjacent to the local site given by;

$$P^{eff} = \left\{ \begin{array}{c} 0 \\ -M_{be}^{\Omega^+} \ddot{u}_e^0 - K_{be}^{\Omega^+} u_e^0 \\ M_{eb}^{\Omega^+} \ddot{u}_e^0 + K_{ee}^{\Omega^+} u_e^0 \end{array} \right\}. \tag{14}$$

- In step-II of the algorithm, ground response analysis is performed at desired microzones using as excitation the free-field motion extracted from the data base created during step-I. In the case of a plane wave analysis the calculation of the incoming motion through the solution of the FE-system specified in step-I is unnecessary as the field can be obtained analytically. This computation is performed automatically by **WAVES** in the elemental subroutines used to define the elements along the strip (see fig. 4).

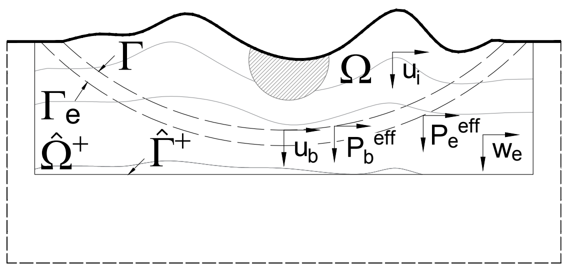


Figure 4: Reduced domain subjected to the action of effective forces P^{eff} and equivalent to the seismic sources.

The local model can take into account a highly detailed description of the small-scale topography. This local step can be performed using computing resources typically available at a consulting office and using an independent numerical technique.

A key aspect in the DRM technique is the fact that in the reduced domain $\hat{\Omega}^+ \cup \Omega$, the domain $\hat{\Omega}^+$ is of the same material properties as in the original Ω^+ (right part of fig. 2). To reflect this fact the mass and stiffness coefficients pertaining to the domain $\hat{\Omega}^+$ have retained the superscript Ω^+.

3 Explicit solution scheme

Consider the discrete dynamic equilibrium equations at time t

$$M^t A + C^t V + K^t U =^t F \tag{15}$$

and where M, C, K are the assembled mass, damping and stiffness matrices respectively while ${}^tA$, ${}^tV$, ${}^tU$ and ${}^tF$ are the nodal accelerations, velocities, displacements and external loads vectors at time t. In **WAVES** the external forces may be the result of particular point loads or effective forces consistent with a plan wave analysis.

In terms of nodal forces, eq. (15) can be written like;

$${}^tF^I +^t F^D +^t F^s =^t F \tag{16}$$

where ${}^tF^I$, ${}^tF^D$ and ${}^tF^s$ are inertial, damping, elastic and external nodal forces respectively.

Expanding the acceleration and velocity terms at time t in a consistent finite central differences scheme we have;

$$\begin{aligned} {}^tA &= \frac{1}{\Delta t^2}\left({}^{t-\Delta t}U - 2{}^tU +^{t+\Delta t} U\right) \\ {}^tV &= \frac{1}{2\Delta t}\left(-{}^{t-\Delta t}U +^{t+\Delta t} U\right). \end{aligned} \tag{17}$$

Consider now the trial states

$$\begin{aligned} {}^t\hat{A} &= \frac{1}{\Delta t^2}\left({}^{t-\Delta t}U - 2{}^tU\right) \\ {}^t\hat{V} &= -\frac{1}{2\Delta t}{}^{t-\Delta t}U \end{aligned} \tag{18}$$

which allows us to write eq. (17) like;

$$\begin{aligned} {}^tA &= {}^t\hat{A} + \frac{1}{\Delta t^2}{}^{t+\Delta t}U \\ {}^tV &= {}^t\hat{V} + \frac{1}{2\Delta t}{}^{t+\Delta t}U \end{aligned} \tag{19}$$

Notice that the terms ${}^t\hat{A}$ and ${}^t\hat{V}$ result after assuming that ${}^{t+\Delta t}U = 0$ in eq. (17) thus they are commonly referred to as predictors. Notice also that after the displacements at $t + \Delta t$ have been found the corrected values of ${}^tA$ and ${}^tV$ can be obtained via eq. (19). In this sense the terms $\frac{1}{\Delta t^2}{}^{t+\Delta t}U$ and $\frac{1}{2\Delta t}{}^{t+\Delta t}U$ play the role of correctors to the initial predictors. The resulting algorithm is commonly referred to, by obvious reasons, as a predictor-corrector scheme.

Using eq. (19) in eq. (15) results in the following equation governing the displacements at time $t + \Delta t$;

$$\left(\frac{1}{\Delta t^2}M + \frac{1}{2\Delta t}C\right)^{t+\Delta t} U =^t F - M^t\hat{A} - C^t\hat{V} - K^tU \tag{20}$$

The dynamic finite element equilibrium equations given in eq. (20) can be written in the standard static form:

$$\hat{K}^{t+\Delta t}U = {}^t\hat{F}$$

after letting;

$$\hat{K} = \frac{1}{\Delta t^2}M + \frac{1}{2\Delta t}C$$

and

$${}^t\hat{F} = {}^tF - M^t\hat{A} - C^t\hat{V} - K^tU$$

.

It is convenient, and physically appealing, to write eq. (20) in terms of forces like:

$${}^{t+\Delta t}F^I +^{t+\Delta t} F^D =^t F -^t \hat{F}^I -^t \hat{F}^D -^t F^s \tag{21}$$

- Equation (21) is an equilibrium equation at time $t = t$ allowing to predict the displacements at time $t = t + \Delta t$ in terms of previously known values at times t and $t = t - \Delta t$ as schematically shown in fig. 5

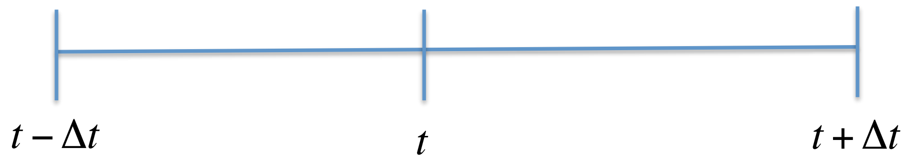


Figure 5: Definition of the general iteration giving displacements at $t = t+\Delta t$ in terms of previously known values at times t and $t = t - \Delta t$.

- The equation is exact within the error introduced by the expansion used in eq. (17).
- The first predicted solution is at $t = \Delta t$ which implies that we require data at $t = -\Delta t$ and at $t = 0$ (i.e., initial conditions) as schematically shown in fig. 6.

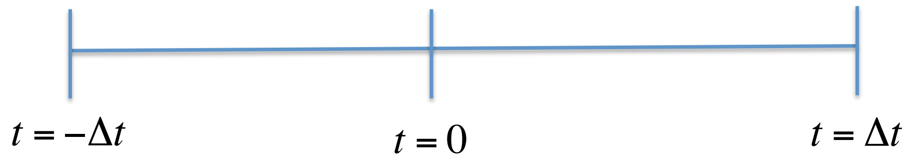


Figure 6: Initial iteration predicting values at time $t = \Delta t$ in terms of the artifitial values at $t = -\Delta t$ and the initial conditions at $t = 0$.

Damping Assumptions

The damping matrix C appearing in eq. (20) is actually not assembled in the standard finite element sense but it is rather built from the mass and stiffness matrices. Moreover we can consider the following possibilities regarding damping.

(i) Neglect damping (This is however inconvenient for finite domains) which gives:

$$^{t+\Delta t}F^I +^{t+\Delta t} F^D =^t F -^t \hat{F}^I -^t F^s \tag{22}$$

(ii) Use Rayleigh Damping and retain the velocity expansion used in (17). That is;

$$C = \alpha M + \beta K \tag{23}$$

then we have (in terms of forces);

$$(1 + \beta \Delta t^2)^{t+\Delta t}F^I + \frac{\alpha}{2\Delta t}{}^{t+\Delta t}F^S =^t \hat{F} \tag{24}$$

where;

$$^t\hat{F} =^t F -^t \hat{F}^I -^t \hat{F}^D -^t F^s$$

Solution in equation eq. (24) requires the full assembly and factorization of an effective stiffness matrix.

(iii) Use Rayleigh damping but modify the velocity expansion introduced in eq. (17). Using

$$^tV = \frac{1}{\Delta t}(^tU -^{t-\Delta t} U) \tag{25}$$

yielding;

$$^{t+\Delta t}F^I =^t F -^t \hat{F}^I -^t \hat{F}^D -^t F^s \tag{26}$$

where now the velocity predictor and corrector terms are defined like:

$$^t\hat{V} = \frac{1}{\Delta t}\left(^tU - {}^{t-\Delta t}U\right)$$

and

$$^tV = \frac{1}{2\Delta t}\left(-{}^{t-\Delta t}U + {}^{t+\Delta t}U\right)$$

respectively, giving the equation:

$$^{t+\Delta t}F^I =^t F -^t \hat{F}^I -^t F^s -^t \hat{F}^D \tag{27}$$

Algorithm implemented in WAVES (damping assumption 3)

Let us write eq. (27) like

$$^{j+1}F^{I} =^{j} F -^{t} \hat{F}^{I} -^{j} F^{s} -^{j} \hat{F}^{D}. \tag{28}$$

The first prediction corresponding to $t = \Delta t$ is given by;

$$^{\Delta t}F^{I} = {}^{0}F - {}^{0}\hat{F}^{I} - {}^{0}\hat{F}^{S} - {}^{0}\hat{F}^{D}$$

which results after applying eq. (27) at $t = 0$ and where the term

$$^{0}\hat{F}^{D} = C\frac{1}{\Delta t}\left({}^{0}U - {}^{-\Delta t}U\right)$$

requires knowing the fictitious value $-^{\Delta t}U$. This last term can be computed after applying the central difference expansion at $t = 0$ and solving for $-^{\Delta t}U$ giving;

$$^{-\Delta t}U =^{0} U - \Delta t^{0}V + \frac{\Delta t^{2}}{2}{}^{0}A \tag{29}$$

Finally using eq. (29) in eq. (28) allows us to start up the algorithm. The general and initial iteration are schematized in fig. 7 and fig. 8 respectively.

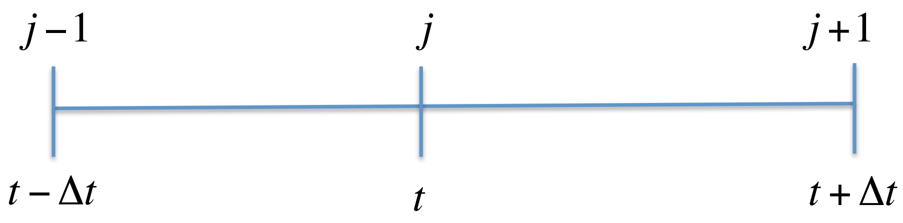


Figure 7: Definition of the general iteration

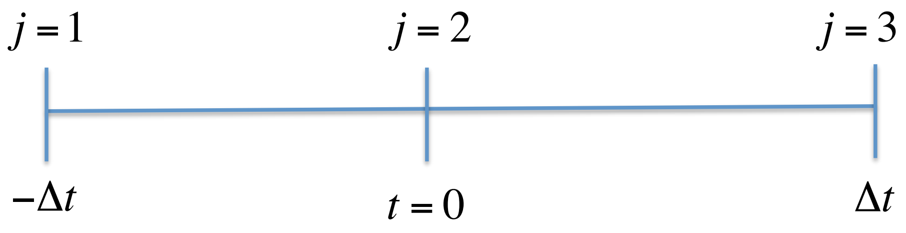


Figure 8: Definition of the initial iteration

3.1 Decoupling

The equilibrium equations discussed so far can be decoupled whenever the coefficient matrix is diagonal. As a result, the algorithm can proceed one degree of freedom at a time without the need for a complete assembly process and with large memory savings. Consider the particular case of damping assumption 3 and a lumped mass matrix. Writing eq. (28) for the i-th degree of freedom (i.e., i is kept fixed) results in:

$$\frac{1}{\Delta t^2}M_{ij}^{t+\Delta t}U_j =^t F_i - K_{ij}{}^tU_j - M_{ij}{}^t\hat{A}_j - C_{ij}{}^t\hat{V}_j \tag{30}$$

Now, if the lumped mass matrix is written like;

$$M_{ij} = m_I\delta_{ij}$$

we have:

$$\frac{1}{\Delta t^2}m_I^{t+\Delta t}U_i =^t F_i - K_{ij}{}^tU_j - m_I{}^t\hat{A}_i - C_{ij}{}^t\hat{V}_j \tag{31}$$

producing the following recursive equation;

$${}^{t+\Delta t}F_i^I =^t F_i -^t \hat{F}_i^S - {}^t\hat{F}_i{}^I - {}^t\hat{F}_i{}^D. \tag{32}$$

To initialize the algorithm we apply the FDs equations at $t = 0$ leading to:

$$\frac{1}{\Delta t^2}m_I^{\Delta t}U_i =^0 F_i - K_{ij}{}^0U_j - m_I{}^0\hat{A}_i - C_{ij}{}^0\hat{V}_i.$$

In the above we require:

$${}^0\hat{V} = \frac{1}{\Delta t}\left({}^0U_i - {}^{-\Delta t}U_i\right)$$

which at the same time implies prior knowledge of the artificial term ${}^{-\Delta t}U_i$. This can be obtained from (29) as follows:

$${}^{-\Delta t}U_i =^0 U_i - \Delta t^0V_i + \frac{\Delta t^2}{2}{}^0A_i. \tag{33}$$

The initial acceleration is obtained after assuming homogeneous ICs;

$$m_I{}^0A_i + C_{ij}{}^0V_j + K_{ij}{}^0U_j =^0 F_i$$

therefore

$$m_I^0A_i = \frac{{}^0F_i}{m_I}$$

and

$${}^{-\Delta t}U_i = \frac{\Delta t^2}{2m_I}{}^0F_i.$$

Moreover, neglecting the damping effects on the prediction of ${}^{\Delta t}U_i$ yields;

$${}^{\Delta t}U_i = \frac{\Delta t^2}{2m_I}{}^0F_i$$

Algorithm 1: Full Algorithm

```
Data: Time span, Geometry, Material Parameters
Result: Displacements, Velocity and Acceleration time histories
Initialize solution vectors ($j = 1$ corresponds to ICs: 0 superscript indicates $t = 0$);
${}^{0}U_i \longleftrightarrow {}^{j=1}U_i = 0,\ {}^{0}V_i = 0,\ {}^{j=1}A_i = \frac{{}^{1}R_i}{m_I}$ ;
Select $\Delta t$ and damping coefficients $\alpha$, $\beta$;
Fix 1-st predicted value (let $j = 2$);
    $${}^{\Delta t}U_i \longleftarrow \frac{\Delta t^2}{2m_I}{}^{0}F_i \longleftrightarrow \left[{}^{j=2}U_i \longleftarrow \frac{\Delta t^2}{2m_I}{}^{j=1}F_i\right]$$
Time Integration Phase;
while $j \le N$ do
    Predictors
    ${}^{j}\hat{A}_i \longleftarrow \frac{1}{\Delta t^2}\left({}^{j-1}U_i - 2{}^{j}U_i\right)$
    ${}^{j}\hat{V}_i \longleftarrow \frac{1}{\Delta t}\left({}^{j}U_i - {}^{j-1}U_i\right)$
    ${}^{j+1}F_i^I \longleftarrow {}^{j}F_i - K_{ij}{}^{j}U_j - m_I{}^{j}\hat{A}_j - C_{ij}{}^{j-1}\hat{V}_j$
    Solve
    ${}^{j+1}U_i \longleftarrow \frac{\Delta t^2}{m_I}{}^{j+1}F_i^I$
    Correctors
    ${}^{j}A_i \longleftarrow {}^{j}\hat{A}_i + \frac{1}{\Delta t^2}{}^{j+1}U_i$
    ${}^{j}V_i \longleftarrow {}^{j}\hat{V}_i + \frac{1}{2\Delta t}{}^{j+1}U_i$
    $j \longleftarrow j + 1$
end
```

Program structure

In the explicit integration scheme the solution corresponding to the i-th degree of freedom at time $t + \Delta t$ is found from eq. (34):

$$ {}^{j+1}U_i \longleftarrow \left(\frac{\Delta t^2}{m_I}\right){}^{j+1}{F^I}_i \tag{34}$$

where m_I is the total mass associated to the i-th degree of freedom and ${}^{j+1}F_i^I$ is the total force considering the contribution from the external, elastic, inertial and damping terms at times t and $t - \Delta t$. This force is given by eq. (35):

$$ {}^{j+1}F_i^I \longleftarrow {}^{j}F_i - K_{ij}{}^{j}U_j - m_I{}^{j}\hat{A}_i - C_{ij}{}^{j}\hat{V}_j. \tag{35}$$

Accordingly, in the uncoupled explicit finite element formulation the equation solving process proceeds one degree of freedom at a time. This implies a different assembly process to the one used in an implicit algorithm where a formal coefficient matrix is assembled and inverted. Now the mass, damping and stiffness elemental matrices are used to obtain effective nodal forces at each degree of freedom. In summary the mesh is not covered in an element by element basis, but in a node by node basis. In the following algorithm we discuss this nodal assembly process where in order to solve the displacement at a given degree of freedom prior knowledge of the elements contributing to the given node is necessary.

In particular in **WAVES** the total mass m_I and the total force ${}^{j+1}F_i^I$ is built after considering the contribution from the different elements connected to the node. This nodal assembly process is schematized in fig. 9 in which 4 bi-lineal elements are connected by a central node.

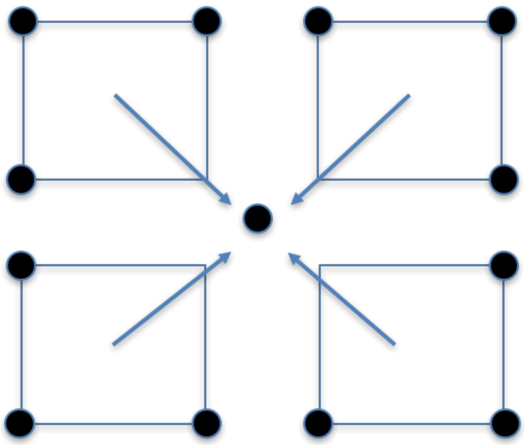


Figure 9: Nodal assembly

The nodal assembly is described in the following algorithm:

Algorithm 2: Nodal assembly

```
Data: Number of elements connected to the node
Result: Total mass and force contribution
for $i \longleftarrow 1$ to $Numnp$ do
    $k \longleftarrow NIEL_i$
    for $j \longleftarrow 1$ to $k$ do
        Retrieve element parameters
        Compute predictors
        Compute element contribution (UEL.for)
        Assembly element contribution into the force vector ${}^{j+1}F_i^I$
    end
    Solve for the current d.o.f
    Perform corrections.
end
```

The subroutine **UEL.for** in the algorithm computes the contribution from the current element to the degree of freedom being solved. The library of available elements in the code corresponds precisely to a set of **UELs** subroutines. Additional elements can be easily implemented and added to the code by just coding new **UEL** subroutines as will be specified later.

Elements

The following elements are currently available in **WAVES** :

```
UELEXP8: 8-noded 2D quad element.
UELEXP4: 4-noded 2D quad element.
UELEXP9: 9-noded 2D quad element.
UELEXP6: 6-noded 2D triangular element.
UEL8INCOT: 8-noded 2D quad element with plane-wave incoming effective DRM forces.
UEL9INCOT: 9-noded 2D quad element with plane-wave incoming effective DRM forces.
UELEDASH3: 3-noded 2D Lysmer and Kuhlemeyer absorbing boundary.
UELDASHINC: 3-noded 2D transmitting boundary.
UEL3DEXP8: 8-noded bilineal 3D tetrahedarl elment.
```

3.2 Adding elements

User element subroutines may be added to the code by implementing a **UEL** subroutine with the following interface.

```
      SUBROUTINE UELXXX(RHS,ANMASS,NDOFEL,PROPS,NPROPS,IPROPS,NIPROPS,
     1                  AWAVE,IWAVE,COORDS,MCRD,NNODE,U,DU,V,A,JELEM,
     2                  DT,KINC,NINCR)

      IMPLICIT REAL*8(A-H,O-Z)

      PARAMETER (ZERO=0.D0,HALF=0.5D0,ONE=1.D0,NTENS=4,TWO=2.D0,NGPTS=9,
     1           THREE=3.D0, NDI=3)

C     Parameter arrays from UEL.f

      DIMENSION RHS(NDOFEL),ANMASS(NDOFEL),PROPS(NPROPS),
     1          IPROPS(NIPROPS),AWAVE(5),COORDS(MCRD,NNODE),U(NDOFEL),
     2          DU(NDOFEL),V(NDOFEL),A(NDOFEL),AMATRX(NDOFEL,NDOFEL),
     3          AMASS(NDOFEL,NDOFEL),CDMAT(NDOFEL,NDOFEL),UP(NDOFEL),
     4          DDSDDE(NTENS,NTENS),B(NTENS,NDOFEL),BT(NDOFEL,NTENS),
     5          FRST2(NDOFEL,NDOFEL),FRST1(NDOFEL,NDOFEL),XP(2,NGPTS),
     6          XW(NGPTS),AUX1(NTENS,NDOFEL)



      RETURN
```

The subroutine must return the following parameters to the main program:

- RHS: Contribution from the current element to the i-th degree of freedom total external force as given by eq. (35).
- ANMASS: Contribution from the current element to the i-th degree of freedom total mass.

On the other hand the input parameters are defined as follows:

- NDOFEL: Total number of degrees of freedom of the elemnt.
- PROPS: Array containing the real material properties needed to compute the forces in the current element.
- NPROPS: Dimension of the PROPS array.
- IPROPS: Array containing the integer material properties needed to compute the forces in the current element. This arrays is mainly used in the definition of incoming elements in order to specify which face of the element is in direct contact with the scatterer.
- NIPROPS: Dimension of the IPROPS array.
- AWAVE: Plan wave parameters.
- IWAVE: Dimension of the AWAVE array.
- COORDS: Array with the nodal coordinates of the element.
- MCRD: Parameter defining the dimensionality of the problem.
- NNODE: Total number of nodes in the element.
- U: Nodal displacements at time t.
- DU: Displacement increment defined as $\Delta U = {}^{t}U - {}^{t-\Delta t}U$.
- V: Trial velocity defined as ${}^{t}\hat{V} = -\frac{1}{2\Delta t}{}^{t-\Delta t}U$.
- A: Trial acceleration defined as ${}^{t}\hat{A} = \frac{1}{\Delta t^2}\left({}^{t-\Delta t}U - 2{}^{t}U\right)$.
- JELEM: Current element identifier.
- DT: Size of the time step.
- KINC: Current increment number.
- NINCR: Total number of increments.

4 Modeling a wave propagation problem with WAVES

In **WAVES** a finite element model is defined through an input data file with extension **.inp** as described in the file named **template.inp**. The file contains the following general data blocks.

```
PROBLEM DEFINITION BLOCK
NODAL DEFINITION BLOCK
MATERIAL DEFINITION BLOCK
WAVE DEFINITION BLOCK
ELEMENTS DEFINITION BLOCK
POINT LOADS DEFINITION BLOCK
NODAL OUTPUT DEFINITION BLOCK
```

A template file is provided with the code. The structure of the input data file is explained in the last section of this document.

Creating a model with Gmesh

Although input data files for simple models can be created manually the process becomes cumbersome in large problems. In this section we describe how to define a model using the third party software **Gmesh**. On the other hand, although the code can be used to solve virtually any dynamic loading problem, in the following example we focus in a plane wave analysis typically found in earthquake engineering.

The creation of a **WAVES** model in **Gmesh** involves a three-step procedure:

(i) Definition of the model geometry through a **Gmsh *.geo** file.

(ii) Definition of the finite element mesh through a **Gmsh *.msh** file.

(iii) Definition of the input data file ***.inp** using the python module **meshio.py**.

Problem definition

All the plane wave propagation problems in **WAVES** can be defined starting from the fundamental geometry shown in fig. 10 given by an open rectangular box. This fundamental geometry represents a half-space or baserock used for the application of the plane incident waves. A plane wave scattering model can subsequently be created through the addition of a generalized geometry inside the rectangular box. The figure shows the pre-defined physical lines and surfaces as required by Gmsh and used later during the generation of the input file. The model is ahown upsidedown as originally created in Gmsh in order to preserve the typical wave propagation convention of having the y-axis pointing towards the interior of the half-space. Figure 11 shows the model corresponding to a scatterer conformed by triangular canyon. In terms of Gmsh entities the scatterer is conformed by the addition of the physical surface 12000 and the lines 22 through 25 configuring the canyon.

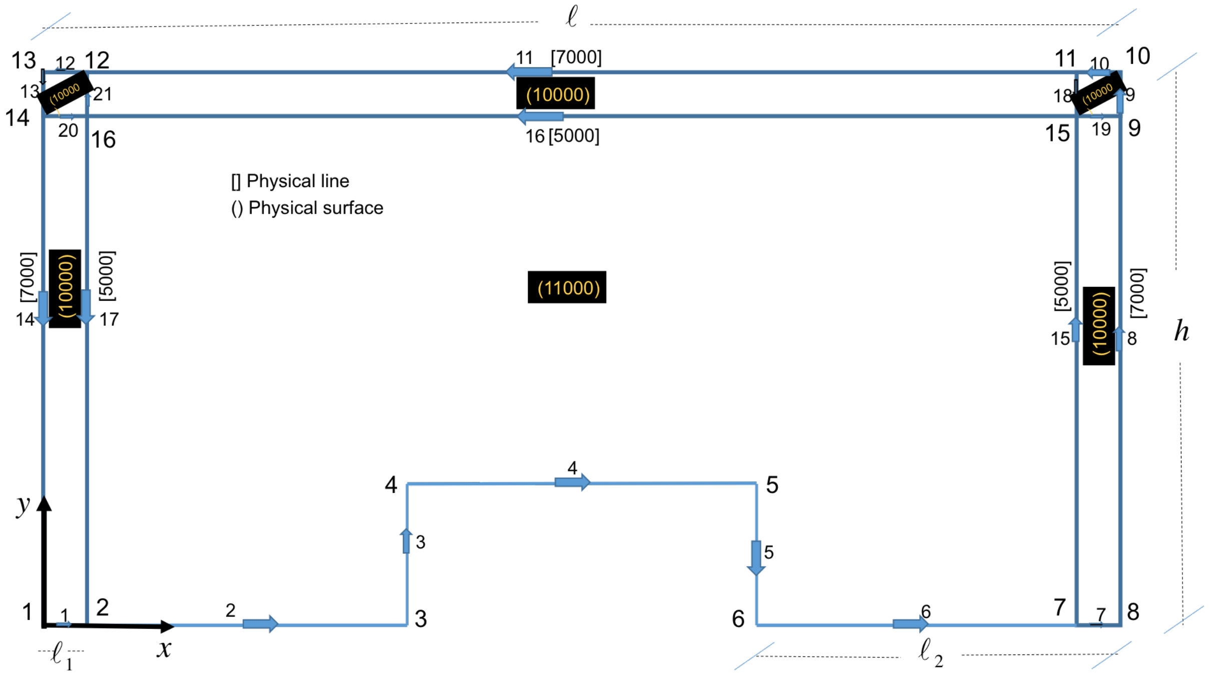


Figure 10: Geometry of the general template conformed by a rectangular box used to represent the half-space.

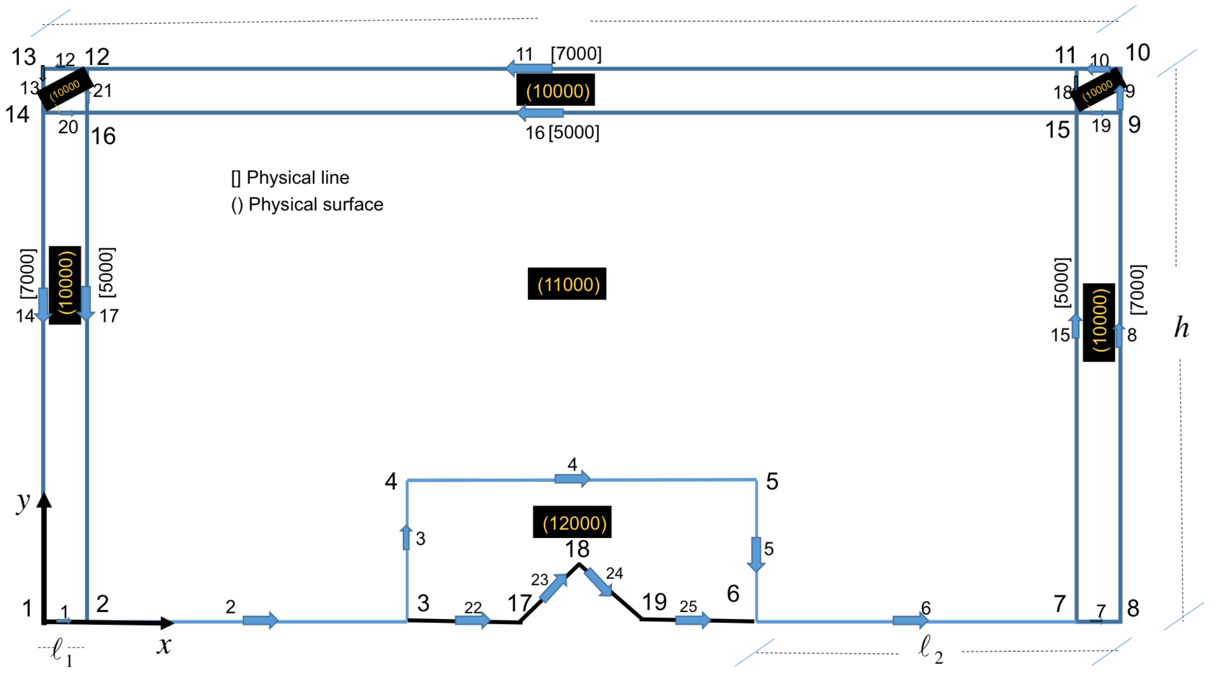


Figure 11: Particular scatterer in the form of a V-shaped canyon added to the rectangular box.

Meshing the model

To conduct the wave propagation analysis using **WAVES** the **Gmsh** model must be converted into the **.inp** data file. This reading and re-writing process is achieved using the Python module **meshio** which directly reads the **Gmsh** file into dictionaries which are later processed by problem-dependent python scripts. The conversion of a mesh into a **WAVES** input file involves the generation of arrays storing the nodal and elementary data for the different parts of the model. A **Gmsh** model is converted into a **WAVES** model after processing the nodal data and each physical surface and physical line of the finite element model. This process is summarized as follows:

- Reading the **Gmsh** data files using meshio.
- Storing the nodal data in a nodal array written in **WAVES** format.
- Process each physical surface of the model and generate elementary arrays in **WAVES** format.
- Process the physical line modeling the absorbing boundary condition and generate an additional elementary array.
- If any process additional physical lines to impose point loads and point loads array.
- Write all the arrays into a single **WAVES** file.

These steps can be conducted using simple subroutines from the module mesh_waves.py available in the folder **MODELS>MESHER**. Each one of these subroutines is described next.

Nodal writer

```
def node_writer(points , point_data):
    """
    Writes down the nodal data as required by WAVES.
        INPUT PARAMTERS:
    ---------------
        points and point_data: Are the dictionaries creatd by meshio.
    OUTPUT PARAMTERS:
    ----------------
        nodes_array : Integer array with the nodal data according to WAVES.
    """
```

Takes as input parameters the dictionaries **points** and **point_data** storing the nodal coordinates and nodal identifiers of nodal sets associated to physical entities. The subroutine returns as output parameter the array **nodes_array** ready to be written into the **WAVES** input file.

Element writer

Process each physical surface containing elements of the same type and of the same material profile. The subroutine takes as input parameters the dictionaries **cells** and **cell_data** and the name of the physical surface where they belong. Upon execution the subroutine returns the number of elements contained in the set and an array with the elemental data written in **WAVES** format. The subroutine must be called one per each physical surface declared in the model. In order to keep a consecutive count in the number of elements the subroutine uses input and output parameters **nini** and **nf**. The former initializes the element count and the latter indicates the last element number on the set. When the subroutine is used to process consecutive physical surfaces the parameter **nini** must be initialized with the last computed value of **nf**.

```
def ele_writer(cells , cell_data , ele_tag , phy_sur , ele_type , mat_tag , ndof ,
                      nnode , nini):
    """
    Extracts a subset of elements from a complete mesh according to the
    physical surface phy_sur and writes down the proper fields into an
    elements array.
    INPUT PARAMTERS:
    ---------------
        cell and cell_data: Are the dictionaries creatd by meshio.
        ele_tag : String defining the element type according to meshio
        (e.g., quad9 , line3, etc).
        phy_sur : Integer defining the physical surface for the subset.
        ele_type: Integer defining the element type according to WAVES.
        mat_tag : Integer defining the material profile for the subset.
        ndof    : Integer defining the number of degrees of freedom for the elements.
        nnode   : Integer defining the number of nodes for the element.
        nini   : Integer defining the element id for the first element in the set.
    OUTPUT PARAMTERS:
    ----------------
        nf        : Integer defining the element id for the last element in the set
        els_array : Integer array with the elemental data according to WAVES.
    """
```

Face recognition subroutine

This subroutine is a special version of the element writer subroutine used to process the incoming elements containing the strip where the incident plane wave is applied in the form of effective forces through stiffness terms. In order to compute the effective loads these elements require specification of the normal vector of the element face in direct contact with the scatterer. The subroutine recognizes the corresponding element face by identifying the element nodes in contact with a specified physical line defining the halfspace-scatterer interface. The material profiles with identifiers 1 through 4 will then have an intger material property corresponding to the possible 4 element faces. Accordingly material profile 4 in the input data file will be assigned to all the incoming elements with face number 4 in contact with the scatterer-half-space physical line.

```
def face_recognition(cells , cell_data , phy_sur , phy_lin , nini):
    """
    For the incoming elements required to create a plane wave
    this function extracts the set according to the physical
    surface and identifies the element face in contact with
    th scatterer.
    INPUT PARAMTERS:
    ---------------
        cell and cell_data: Are the dictionaries creatd by meshio
        phy_sur: Integer defining the physical surface for the strip.
        phy_lin: Integer defining the physical line for the strip-scatterer interface
        nini   : Integer defining the element id for the first element in the set.
    OUTPUT PARAMTERS:
    ----------------
        nf        : Integer defining the element id for the last element in the set
        els1_array: Integer array with the elemental data according to WAVES.
    """
```

Absorbing boundary writer subroutine

The Lysmer and Khulemeyer absorbing boundaries are specified in terms of a physical line in **Gmsh**. The absorbing boundary writer subroutine identifies which line elements out of the **cell_data** dictionary belong to this physical line and subsequently write the specific element data in **WAVES** format.

```
def boundary_writer(cells , cell_data , phy_lin , mat_tag , nini ):
    """
    Extracts a subset of line elements from a complete mesh and corresponding
    to Lysmer absorbing  boundaries. The elements are identified according to
    the physical line phy_lin.
    INPUT PARAMTERS:
    ---------------
        cell and cell_data: Are the dictionaries created by meshio.
        ele_tag : String defining the element type according to meshio
        (e.g., quad9 , line3, etc).
        phy_lin: Integer defining thw physical line for the strp-scatterer
        interface.
        mat_tag : Integer defining the material profile for the subset.
        nini   : Integer defining the element id for the first element in the set.
    OUTPUT PARAMTERS:
    ----------------
        nf        : Integer defining the element id for the last element in the set
        els_array : Integer array with the elemental data according to WAVES.
    """
```

Example: Creating the model for the generalized template

Here we illustrate how to create a **WAVES** model for the generalized template shown again in fig. 12. In the **template.geo** file in **Gmsh** the model has been divided into physical surfaces and physical lines identifying the different parts of the domain.

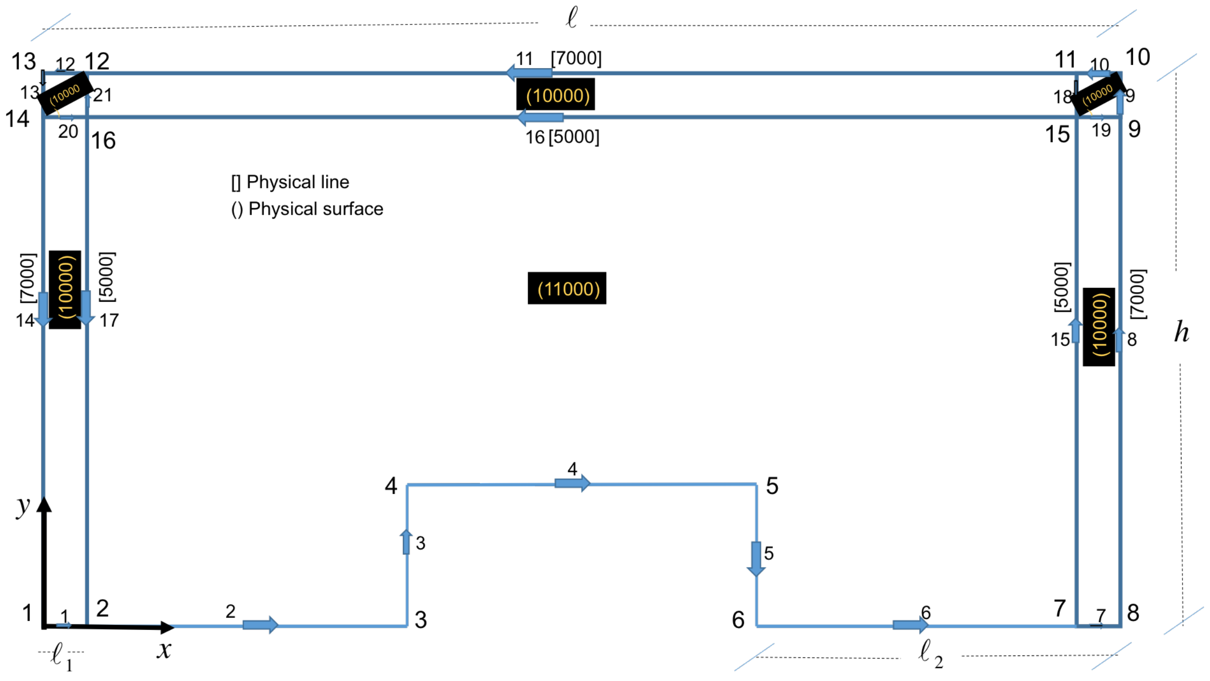


Figure 12: Geometry of the general template conformed by a rectangular box used to represent the half-space.

The generalized scatterer is to be subjected to a vertically incident SV plane wave in the form of a Ricker pulse of central frequency $4.0Hz$, which implies waves of frequencies as large as $16Hz$ are to be propagated. The half-space has a mass density $\rho = 1.0Kg - m/m^3$ and a shear wave propagation velocity of $1.0Km/s$. The characteristic element size is computed as:

$$h_c = \frac{\beta_{\min}}{12 f_{\max}} \tag{36}$$

where β_{min} is the minimum shear wave propagation velocity in the medium and f_{max} is the largest frequency expected to be propagated. Here we will use characteristic element sizes of 10m for the incoming elements and of 25m for the half-space elements. This is specified in the geometry file **template.geo** by the meshing parameters $cl1 = 0.1$ and $cl2 = 0.25$. Notice that all the dimensions in the .geo file are scaled by a factor of 10 in order to avoid accuracy loss in the resulting mesh. The proper units are recovered in the **node_write** subroutine of the processing script.

The analysis time step (which is a highly sensitive parameter in the explicit algorithm) is obtained according to:

$$\Delta t = \frac{h_c}{10\alpha}. \tag{37}$$

where α is the P-wave propagation velocity. This is specified directly in the header block of the input file by selecting the appropriate time window and number of increments of the analysis. In this case the time span is of 2.0s divided in 4000 time steps. In the input file this is specified by the 4001 value where the extra increment is an algorithmic requirement of the explicit scheme.

Th Gmsh .geo file

The geometry file used to define the model is explained next. In **Gmsh** a model is defined in terms of hierarchical geometric objects like points, lines and plane surfaces. In addition physical objects like physical points, physical lines and physical surfaces can be assigned to the geometric objects in order to facilitate the processing of the **Gmsh** file as required by **WAVES**. The first part of the .geo file describes useful parameters in the model:

```
//HEADING BLOCK
//For accuracy all dimensions have been scaled by a factor of 10.0
cl1=0.100; //Characteristic dimension of the incoming elements. DO NOT MODIFY.
cl2=0.250; //Characteristic dimension for the elements along other regions
of the domain.
l = 15.0; //Total domain length.
h =  7.5; //Total domain height.

l1=0.20; //Width of the strip. DO NOT MODIFY. THE CODE IS HIGHLY SENSITIVE
 TO THIS PARAMETER
l2=1.00; //Length of the additional auxiliary line used to decrease the number
of elements.
```

The next data block defines points required to define the geometry. The fourth parameter in each point definition fixes the target dimension for the elements located in the lines formed by these points.

```
//Points
Point(1 ) = {0   , 0   , 0, cl1};
Point(2 ) = {l1  , 0   , 0, cl1};
Point(3 ) = {l2  , 0   , 0, cl2};
Point(4 ) = {l2  , h-l2, 0, cl2};
Point(5 ) = {l-l2, h-l2, 0, cl2};
Point(6 ) = {l-l2, 0   , 0, cl2};
Point(7 ) = {l-l1, 0   , 0, cl1};
Point(8 ) = {l   , 0   , 0, cl1};
Point(9 ) = {l   , h-l1, 0, cl1};
Point(10) = {l   , h   , 0, cl1};
Point(11) = {l-l1, h   , 0, cl1};
Point(12) = {l1  , h   , 0, cl1};
Point(13) = {0   , h   , 0, cl1};
Point(14) = {0   , h-l1, 0, cl1};
Point(15) = {l-l1, h-l1, 0, cl1};
Point(16) = {l1  , h-l1, 0, cl1};
```

The next hierarchical geometric object correspond to lines which are the union of two or more points. The directional data of each line (see fig. 12) must be taken into account in the definition of the plane surfaces, since this data is used to define the element during the meshing process. In **WAVES** the reference system is located along the free surface of the have space with the y-axis pointing towards the interior of the half-space. Consequently, the elements are defined in clockwise direction. This convention is usually different from the one implicit in static analysis codes where the elements are defined counterclockwise. In order to keep the same convention the finite element model in **Gmsh** is defined upside-down as shown in fig. 12 and the plane surfaces must be also defined counterclockwise. This requirement means that a line encountered during the counterclockwise loop is positive if the loop has the same sense as the line and negative otherwise.

```
//Lines
Line(1)  = {1 , 2 };
Line(2)  = {2 , 3 };
Line(3)  = {3 , 4 };
Line(4)  = {4 , 5 };
Line(5)  = {5 , 6 };
Line(6)  = {6 , 7 };
Line(7)  = {7 , 8 };
Line(8)  = {8 , 9 };
Line(9)  = {9 , 10};
Line(10) = {10, 11};
Line(11) = {11, 12};
Line(12) = {12, 13};
Line(13) = {13, 14};
Line(14) = {14, 1 };
Line(15) = {7 , 15};
Line(16) = {15, 16};
Line(17) = {16, 2 };
Line(18) = {11, 15};
Line(19) = {15,  9};
Line(20) = {14, 16};
Line(21) = {16, 12};
```

The next block defines the plane surfaces. A surface is formed by a loop of lines that is traversed in counterclockwise direction. Each surface is identified by an integer number and the Recombine command is issued in order to form quads out of the union of two triangles.

```
//Surfaces

//(Right strip)
Line Loop(1) = {7, 8, -19, -15};
Plane Surface(1) = {1};
Transfinite Surface {1} Alternated;
Recombine Surface {1};
```

```
// Right strip-lower corner
Line Loop(2) = {9, 10, 18, 19};
Plane Surface(2) = {2};
Transfinite Surface {2} Alternated;
Recombine Surface {2};

//Bottom strip
Line Loop(3) = {-16, -18, 11, -21};
Plane Surface(3) = {3};
Transfinite Surface {3} Alternated;
Recombine Surface {3};


//Left strip-lower corner
Line Loop(4) = {12, 13, 20, 21};
Plane Surface(4) = {4};
Transfinite Surface {4} Alternated;
Recombine Surface {4};

//Left strip
Line Loop(5) = {1, -17, -20, 14};
Plane Surface(5) = {5};
Transfinite Surface {5} Alternated;
Recombine Surface {5};

//Domain enclosed by the strip
Line Loop(6) = {2, 3, 4, 5, 6, 15, 16, 17};
Plane Surface(6) = {6};
//Transfinite Surface {6} Alternated;
Recombine Surface {6};
```

The final part of the file assigns the physical objects to the different parts of the model. In this case the physical surface (10000) represents the part of the domain where **WAVES** computes only the scattered motion, while in the inside surface identified by the physical surface (11000) **WAVES** computes total motions. The incoming elements will belong to the physical surface (10000) and will have a face in contact with the physical line [5000] representing the scatterer-halfspace interface. The rest of the model is defined by the outer boundary specified by the physical line [7000] where the Lysmer-Khulemeyer absorbing boundaries are to be specified.

```
//--------------------
//PHYSICAL SURFACES
//--------------------


// Strip-including its bottom corners
```

```
Physical Surface(10000) = {1, 2, 3, 4, 5};

//Domain enclosed by the strip
Physical Surface(11000) = {6};


//--------------------
//PHYSICAL LINES
//--------------------

//Absorbing boundaries
Physical Line(7000) = {8, 9, 10, 11, 12, 13, 14};

//Strip-domain contact surface
Physical Line(5000) = {15 , 16 , 17};
```

After creating the **.geo** file the next step is meshing the geometry directly from **Gmsh**. The current example uses second order quads.These must be specified in **Gmsh** right after creating the mesh and just before saving the resulting file. All the files for this model are available on the folder **MODELS/[01]TEMPLATE**. However the processing of the **Gmsh** file in order to write the input data file must be conducted in the folder **MODELS/MESHER**.

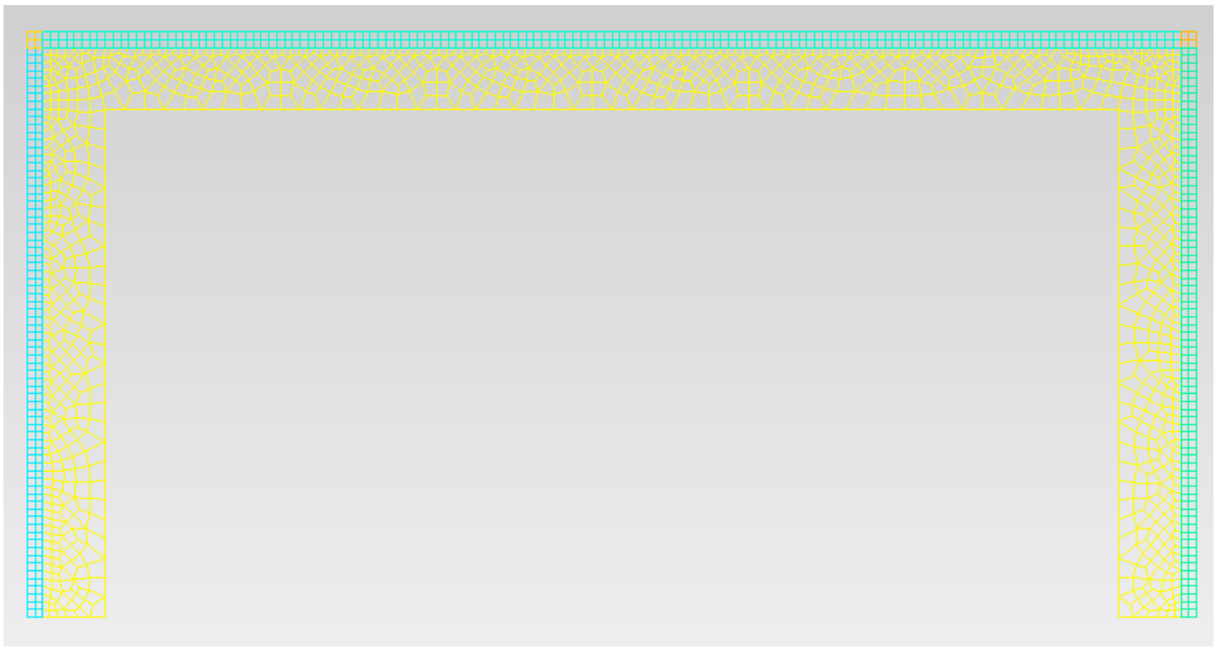


Figure 13: Mesh corresponding to the generalized scatterer (template.msh).

The mesh processing script

As pointed out previously the **Gmsh template.msh** file must be processed by a python script in order to write the **WAVES** input data file. In this specific example the python script **template_input.py**, is used to write the **WAVES** data file **template.inp**. The files can be found in the folder **MODELS/[01]TEMPLATE**.

The heading block

The first part of the script imports the modules **meshio.py** and **mesh_waves.py** required to read the mesh and the previously described subroutines used to write the data file

```
"""
Creates th WAVES.for mesh for a
a generalized scatterer box.
@autor Juan Gomez
"""
from __future__ import division
import meshio
import mesh_waves as msw
import numpy as np
import fileinput
import glob
```

Reading the mesh

In this part of the script the pyhton-written module meshio reads the **Gmsh** file **template.msh** and stores the model into the dictionaries points, cells, point_data, cell_data and field_data.

```
points, cells, point_data, cell_data , field_data = \
    meshio.read("template.msh")
```

Processing the nodes

Now the meshio script creates the array nodes_array and obtains the total number of nodes in the model. The array is already written in **WAVES** format.

```
nodes_array = msw.node_writer(points , point_data)
print len(nodes_array)
```

Processing the elements

The first surface being processed corresponds to the domain representing the part of the half-space behaving as a scatterer and used mainly to propagate the incident wave from the incoming elements. In the model this corresponds to the physical surface (11000). The **ele_writer** subroutine takes as input parameters the python dictionaries storing the mesh, the **Gmsh** element keyword **quad9** identifying the type of elements in the surface, the physical surface tag (11000), the **WAVES** element type code, which in this case corresponds to 9 for quad9 elements, the material profile tag 5, the number of nodes and number of degrees of freedom per element and the initial value of the parameter **nini=0**. After processing the first physical surface the subroutine returns the total number of elements in the set with the parameter **nfin** and the first array of elements array **els1_array**.

The next part of the script takes care of the surface containing the incoming elements. The subroutine is initialized with the final value of the element counter resulting from the previous surface **nini** = **nfin**. The remaining two parameters correspond to the physical surface (10000) and to the physical line [5000] identifying the interface between the half-space and the strip of incoming elements. By default incoming elements are quad9 and this is already considered in the subroutine. In this case the information corresponding to the physical surface (10000) is stored in the elements array **els2_array**.

In the final part of the elements block the script process the absorbing boundary. In this case the routine **boundary_writer** requires as input parameter the initial value of the elements counter **nini** = **nfin**, the physical line identifying the absorbing boundary [7000] and the **WAVES** code 6 identifying the element type. The resulting elements array in **WAVES** format is now stored in **els3_array**.

```
nfin , els1_array  = msw.ele_writer(cells , cell_data , 'quad9' , 11000 , 3 , 5 ,
                                                        18 , 9 , 0 )
nini = nfin
nfin , els2_array  = msw.face_recognition(cells , cell_data , 10000 , 5000 , nini)
nini = nfin
nfin , els3_array  = msw.boundary_writer(cells , cell_data , 7000 , 6 , nini)
```

Writing the template.inp file

In the final part of the script the code writes the nodes and elements arrays into text files and then links all the files into a single .inp file with the name **template.inp**. The numbers 2 trhough 7 in the text files are used to ensure that the final .inp file is assembled in that order. In this case it has been assumed that the user has independently created a file starting by 1, which in this case corresponds to the file **1header.txt** and a file number 3 corresponding to the material block given by the file **3mater.txt**.

```
np.savetxt("2nodes.txt", nodes_array,
           fmt=("%d", "%d", "%d" , "%d" , "%.4f", "%.4f"))
np.savetxt("5eles.txt", els1_array   , fmt="%d")
np.savetxt("6eles.txt", els2_array   , fmt="%d")
np.savetxt("7eles.txt", els3_array   , fmt="%d")


file_list = glob.glob("*.txt")

with open('template.inp', 'w') as file:
    input_lines = fileinput.input(file_list)
    file.writelines(input_lines)
```

Before executing the script the user must create the header and the material files **1header.txt** and **mater.txt**. Th first file contains the general model data while the second file contains the different material profiles for the specific model. The construction of both data blocks is explained in the next section.

After the solution is complete the following solution can be visualized using **Paraview**:

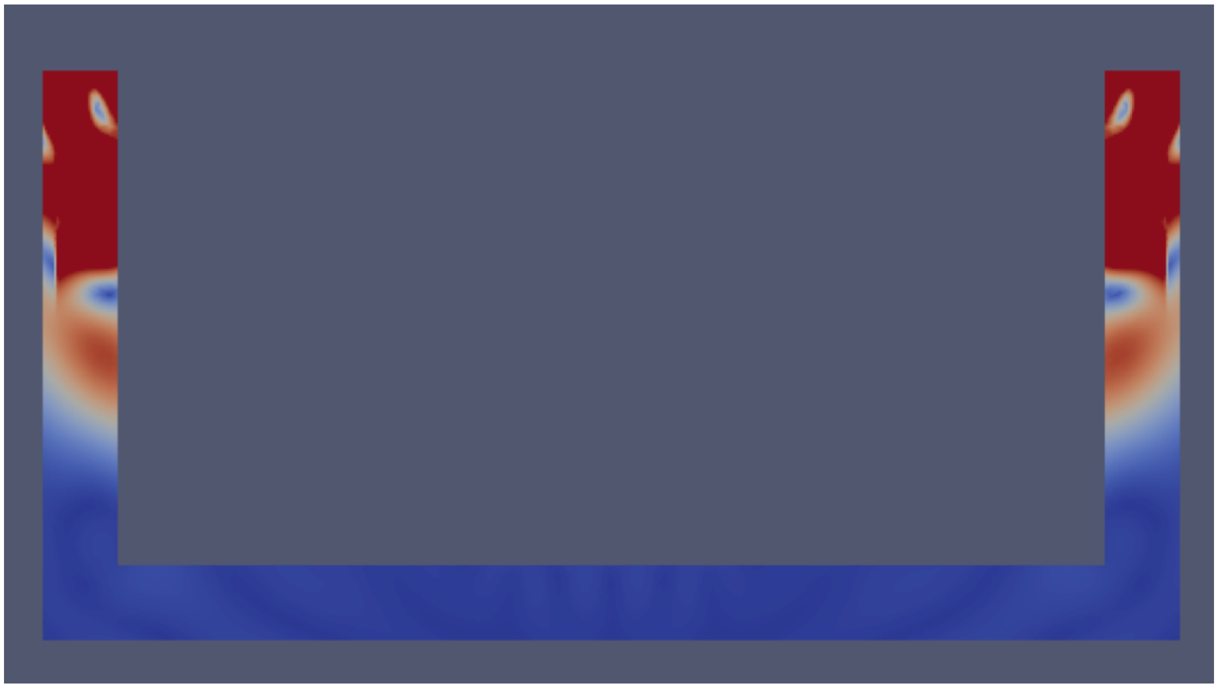


Figure 14: Snapshot of the solution at a particular time instant. The full video is available in the folder **MODELS/[01]TEMPLATE** .

Structure of the *.inp file

This data file is defined as follows:

```
--------------------------------------
------------HEADING BLOCK-------------
--------------------------------------
Title
NNODES     NUMEL   NMAT   Tm  NINCS    NDIM    NMNE    NMDOFE     NMRMAT
NMIMAT     NPLOADS     IMODE    NOUTPU
NSNAP  NPROCESS

NNODES  : Total number of nodes.
NUMEL   : Total number of elements.
NMAT    : Total number of material profiles.
Tm      : Duration of the analysis.
NINCS   : Total number of increments.
NDIM    : Problem dimensionality.
```

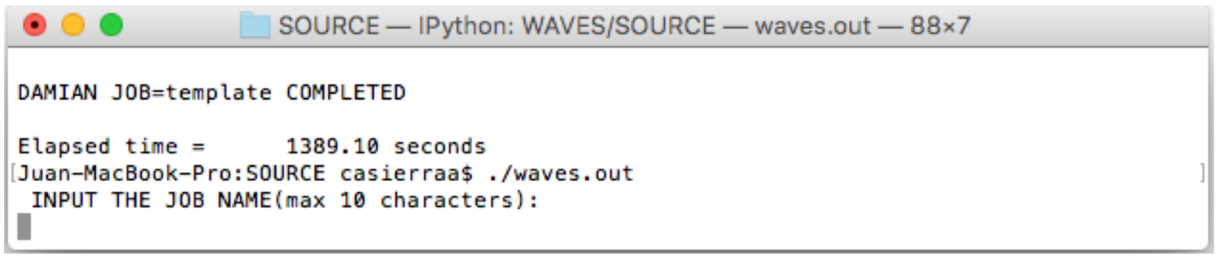
```
NMNE    : Maximum number of nodes per element.
NMDOFE  : Maximum number of degrees of freedom per element.
NMRMAT  : Maximum number of real material properties in a given profile.
NMRMAT  : Maximum number of integer material properties in a given profile.
NPLOADS : Number of point loads.
IMODE   : Point load application mode (0 for Ricker pulse; 1 for an input file)
NOUTPU  : Number of points where output is required.
NSNAP   : Snapshot increment for the creation of VTK fils.
NPROCESS: Number of processors.
--------------------------------------
------------NODAL BLOCK---------------
--------------------------------------
As many data lines as specified by NNODES with the following data:
ID_NODE NDOF  BC-1   BC-2....BC-NDOF   X-coord    Y-coord
ID_NODE  :Nodal identifier.
NDOF     :Number of degrees of freedom at the current node.
BC-1   BC-2....BC-NDOF:Boundary condition flag for each one of the NDOF degrees
of freedom (0 free; 1 restrained)
X-coord  :X-coordinate of the current node.
Y-coord  :Y-coordinate of the current node.
----------------------------------------
------------MATERIAK BLOCK--------------
----------------------------------------
As many data lines as specified by NMAT with the following data:
MAT_ID     NMRMAT   NMIMAT    Par-1   Par-2...Par-NRMAT    IPAR-1
IPAR-2...IPAR-NMIMAT
MAT_ID   :Material profile identifier.
NMRMAT   :Number of real material properties for this profile.
NMIMAT   :Number of integer material properties for this profile.
Par-1   Par-2...Par-NRMAT    :Real material properties. (In a classic 2D wave
propagation analysis the parameters are P-wave velocity; SV wave velocity;
mass density; alpha Rayleigh damping parameter; Beta Rayleigh damping
parameter)
IPAR-1   IPAR-2...IPAR-NMIMAT:Integer material properties.
(This is used in the definition of incoming elements
in order to specify the face in contact with the scatterer).
Profiles 1 through 4 are reserved for incoming elements.
----------------------------------------
------------WAVE BLOCK------------------
----------------------------------------
1 data line with the following wave information:
IWAVE     Tm     Tini     fc     Amp    Phi
IWAVE    :Wave type index (1 for SV;2 for P wave).
Tm       :Time span of the analysis in the case of a Ricker pulse.
Tini     :Initial time of the Ricker pulse.
```

```
fc       :Central frequency of the Ricker pulse.
Amp      :Amplitude of the incident wave.
Phi      :Incidence angle of the plane wave.
----------------------------------------
------------ELEMENTS BLOCK--------------
----------------------------------------
As many data lines as specified by NUMEL each one with the following data:
ID_ELE  ELETYPE  NDOFE     MAT_TAG   NNODES   N1   N2   N3....N-NNODES
ID_ELE  :Element id.
ELETYPE :Element type.
NDOFE   :Number of degrees of freedom at the element.
MAT_TAG :Material profile assigned to the element.
NNODES  :Number of nodal points for the element.
N1   N2   N3....N-NNODES: NNODES-nodal points for the element.
----------------------------------------
------------LOADS BLOCK-----------------
----------------------------------------
As many data lines as loaded nodal points.
ID_NODE   ID_DOF
ID_NODE   :Id of the loaded node.
ID_DOF    :Degree of freedom of the point load.
----------------------------------------
------------OUTPUT BLOCK----------------
----------------------------------------
As many data lines as defined by NOUTPU each one indicating
a nodal identifier at which the output history is required.
```

5 Running the program

To execute **WAVES** use the following command at the terminal window:

```
./waves.out
```



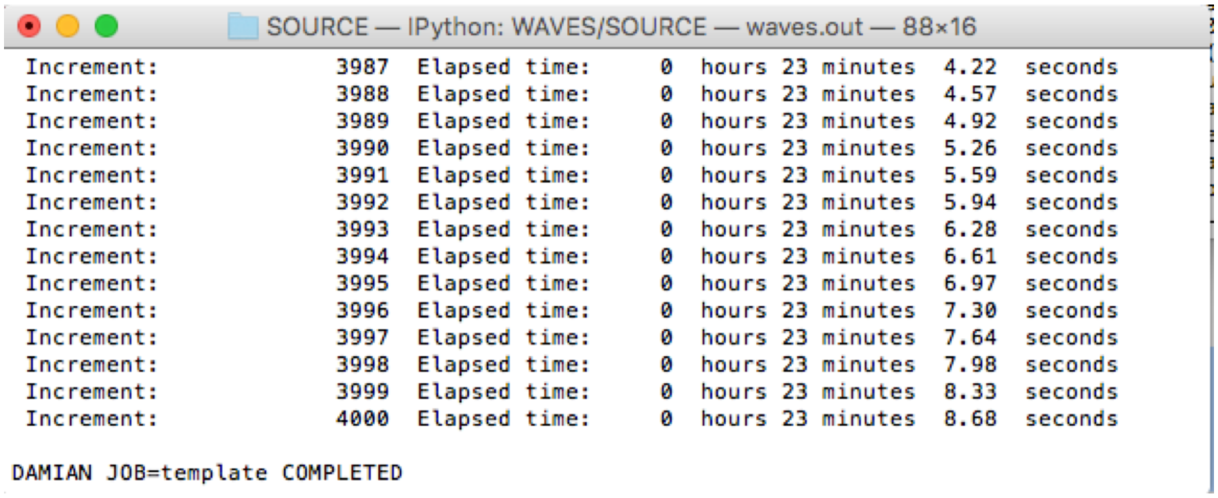
```
SOURCE — IPython: WAVES/SOURCE — waves.out — 88×7

DAMIAN JOB=template COMPLETED

Elapsed time =      1389.10 seconds
Juan-MacBook-Pro:SOURCE casierraa$ ./waves.out
 INPUT THE JOB NAME(max 10 characters):
```

Figure 15: Running WAVES from the terminal window.

and input the name of the data file (without extension). If the analysis is finished successfully you will get the following output:



```
 Increment:           3987  Elapsed time:     0  hours 23 minutes  4.22  seconds
 Increment:           3988  Elapsed time:     0  hours 23 minutes  4.57  seconds
 Increment:           3989  Elapsed time:     0  hours 23 minutes  4.92  seconds
 Increment:           3990  Elapsed time:     0  hours 23 minutes  5.26  seconds
 Increment:           3991  Elapsed time:     0  hours 23 minutes  5.59  seconds
 Increment:           3992  Elapsed time:     0  hours 23 minutes  5.94  seconds
 Increment:           3993  Elapsed time:     0  hours 23 minutes  6.28  seconds
 Increment:           3994  Elapsed time:     0  hours 23 minutes  6.61  seconds
 Increment:           3995  Elapsed time:     0  hours 23 minutes  6.97  seconds
 Increment:           3996  Elapsed time:     0  hours 23 minutes  7.30  seconds
 Increment:           3997  Elapsed time:     0  hours 23 minutes  7.64  seconds
 Increment:           3998  Elapsed time:     0  hours 23 minutes  7.98  seconds
 Increment:           3999  Elapsed time:     0  hours 23 minutes  8.33  seconds
 Increment:           4000  Elapsed time:     0  hours 23 minutes  8.68  seconds

DAMIAN JOB=template COMPLETED
```

Figure 16: Running WAVES from the terminal window.

The displacement solution can be visualized with **PARAVIEW** by opening the response files **Body_Txx.vtk** which are written into the folder **SOURCE/vtk**