ONDES3D: USER GUIDE

(last review: February 22nd 2011)

I. Using the code

A. Launching ONDES3D

The working directory must contain the program files:

- *main.c* and *main.h*
- *alloAndInit.c* and *alloAndInit.h*
- *alloAndInit_LayerModel.c* and *alloAndInit_LayerModel.h*
- *computeIntermediates.c* and *computeIntermediates.h*
- *computeStress.c* and *computeStress.h*
- *computeVeloAndSource.c* and *computeVeloAndSource.h*
- *inelineFunctions.h*
- *IO.c* and *IO.h*
- *nrutil.c* and *nrutil.h*
- *options.h* (the simulation model)
- *struct.h* (the code structure)
- *Makefile* (for compiling)

the files containing the parameters of the simulation:

- *name.prm* (defined in *options.h*)
- *name.map* (defined in *name.prm*)
- *name.hist* (defined in *name.prm*)
- *name_receivers.map* (defined in *name.prm*)

and the files to launch the computation (on Dogger):

- *RUN-MALM-ONDES3D*
- *OARSUB-MALM-ONDES3D*
- *topologie.in*

Several parameters are defined directly in the file *struct.h*:

- the width (number of grid cells) of the absorbing layer:

  ```
  static const int DELTA_VAL = 10;
  ```

- the target reflection coefficient at the absorbing boundary:

  ```
  static const double REFLECT = 0.001;
  ```

- a coefficient used with the option ANLmethod = ANOTHER :

  ```
  static const double f0 = 1.;
  ```

The program is compiled with the file *Makefile*:

```
> make clean
> make
```

which gives the compiled executable program *ondes3d*.

The file *topologie.in* contains two integers: px, number of CPUs in the x direction, and py, number of CPUs in the y direction.

Users need to check in the file *OARSUB-MALM-ONDES3D* if the number of nodes and cores is taken such that the total number of CPUs np is equal to px * py, and to choose the computation duration.

The computation is then launched with:

```
> ./OARSUB-MALM-ONDES3D
```

B. Input files

1. File options.h

This file defines the options of the computation.

```
static const enum typeAnelas ANLmethod = ELASTIC;
```

This option allows choosing if the model behaviour of the material is elastic or anelastic. Possible choices are:

- ELASTIC: elastic law behaviour.
- DAYandBRADLEY: anelastic method of Day and Bradley (2001).
- KRISTEKandMOCZO: anelastic method of Kristek and Moczo (2003).
- ANOTHER: velocities and stresses are computed as in the elastic case and then multiplied by an attenuation function a(q):
  - $a(q) = \exp\left(\frac{-\pi * f0 * dt}{q}\right)$ if $q > 0$
  - $a(q) = 1$ if $q \leq 0$

  where dt is the time step, f0 is defined in the file *struct.h* and q is given in each geological layer in file *name.prm*.

```
static const enum typeInterface interface = USUAL;
```

This option allows choosing how the interface between two horizontal geological layers is described. Possible choices are:

- USUAL: at the interface between two layers, geological parameters (v_p, v_s et ρ) are equal to those of the upper layer.
- KMINTERFACE: the geological parameters are defined at x, y, z by:
  - $\rho(x,y,z) = \int_{x-\frac{dx}{2}}^{x+\frac{dx}{2}} \int_{y-\frac{dy}{2}}^{y+\frac{dy}{2}} \int_{z-\frac{dz}{2}}^{z+\frac{dz}{2}} \rho(u,v,w)\,du\,dv\,dw$

- $$\mu(x,y,z) = \left(\int_{x-\frac{dx}{2}}^{x+\frac{dx}{2}} \int_{y-\frac{dy}{2}}^{y+\frac{dy}{2}} \int_{z-\frac{dz}{2}}^{z+\frac{dz}{2}} \frac{1}{\mu(u,v,w)} dudvdw \right)^{-1}$$
- $$\kappa(x,y,z) = \left(\int_{x-\frac{dx}{2}}^{x+\frac{dx}{2}} \int_{y-\frac{dy}{2}}^{y+\frac{dy}{2}} \int_{z-\frac{dz}{2}}^{z+\frac{dz}{2}} \frac{1}{\kappa(u,v,w)} dudvdw \right)^{-1}$$

following the method of Kristek and Moczo (2003). This option cannot be used when the geological model is read in a file (model = GEOLOGICAL).

static const enum typeSurface surface = FREE;

This option allows choosing if there is a free surface or an absorbing layer above z = ZMAX. Possible choices are:

- ABSORBING: there is an absorbing boundary above z = ZMAX.
- FREE: there is a free surface at z = 0.

static const enum typePML ABCmethod = CPML;

This option allows choosing the type of the absorbing layers at the boundaries of the model. Possible choices are:

- PML: the absorbing layers are PML (Perfectly Matched Layer). This option is possible only with elastic law behaviour (ANLmethod = ELASTIC or ANOTHER).
- CPML: the absorbing layers are CPML (Convolutional Perfectly Matched Layer, e.g. Komatitsch and Martin, 2007).

static const char PRMFILE[50] = "name.prm";

This option gives the name of the parameters file.

static const enum typeModel model = LAYER;

This option allows choosing how the geological model is defined. Possible choices are:

- LAYER: the model is defined with a series of horizontal geological layers which depth and parameters (v_p, v_s et ρ) are given in the file *name.prm*.
- GEOLOGICAL: the geological model is read in a file which name is given in file *name.prm*. This model has the following format:

nx	« number of grid cells in the x direction (west to east) »
ny	« number of grid cells in the y direction (south to north) »
nz	« number of grid cells in the z direction (bottom to top) »
x0	« x coordinate of the model origin (lower southwest corner) »
y0	« y coordinate of the model origin »
z0	« z coordinate of the model origin »
dx	« size of grid cell in the x direction »
dy	« size of grid cell in the y direction »
dz	« size of grid cell in the z direction »

nodata_value out

Then, on the following lines, we find for k = 1 to nz, for j = 1 to ny and for i = 1 to nx, the name of the corresponding geological layer. The names and parameters (v_p, v_s et ρ) of the geological layers are defined in the file *name.prm*.

static const enum typeSource source = HISTFILE;

This option allows choosing how the source function is defined. Possible choices are:

- VELO: the source is a velocity vector introduced directly in the code. At each point source, we write:

$$v_x(i_{source}, j_{source}, k_{source}, t) = v_x(i_{source}, j_{source}, k_{source}, t) - \sin(135 * \pi/180) * 10^7 * 2 * (7\pi)^2 * (t - 1.2/7)\exp\left(-(7\pi)^2 (t-1.2/7)^2\right) * dt/\rho$$

$$v_y(i_{source} + 1/2, j_{source} + 1/2, k_{source}, t) = v_y(i_{source} + 1/2, j_{source} + 1/2, k_{source}, t) - \cos(135 * \pi/180) * 10^7 * 2 * (7\pi)^2 (t - 1.2/7) * \exp\left(-(7\pi)^2 (t-1.2/7)^2\right) * dt/\rho$$

- HISTFILE: the value of source time function is given in the file *name.hist*.

static const enum typeSea sea = SEA;

This option allows taking the sea into account if the geological model is read in a file (model = GEOLOGICAL). Possible choices are:

- NOSEA: the sea is not taken into account. Instead, there is vacuum.
- SEA: the sea is taken into account.

static const enum type:hot snapType = OBOTH;

This option allows choosing which variables are saved into files to draw snapshots. Possible choices are:

- OVELO: velocities on three planes defined in file *name.prm* are saved.
- ODISP: displacements are saved.
- OBOTH: both velocities and displacements are saved.

static const int STATION_STEP = 10;

Values of velocity and stress at the stations are written in files every STATION_STEP time steps (all time steps are saved but the writing operation is done only at some time steps). Files obsN.dat contain:

- the number of the station, its coordinate along x (in m), its coordinate along y and its coordinate along z.
- the number of time steps and the time step.
- for each time step, x, y and z components of the velocity and xx, yy, zz, xy, xz and yz components of the stress.

static const int SURFACE_STEP = 25;

Files to draw snapshots are saved every SURFACE_STEP time steps.

2. Parameter file name.prm

This file follows xml markers. Not every field is required depending of the computation options. The program will read only necessary fields. The computation parameters given in this file are:

- the time step <dt>
- the size of a grid cell <ds>
- the number of time steps <tmax>
- the number of cells in the three direction x, y, z (corresponding to east, north, upward) <xMin>, <xMax>, <yMin>, <yMax>, <zMin> and <zMax>. The model is then a volume of (xMax – xMin) * (yMax - yMin) * (zMax – zMin) points.
- the coordinates of the output planes <i0>, <j0> and <k0>. The program will then write the following files every SURFACE_STEP time steps:
    - o surfacexyvelNNNNNN and surfacexydispNNNNNN contain respectively on five columns x coordinate in km, y coordinate in km and EW, NS and UD velocities or displacements on the plane z = k0 * ds.
    - o surfacexzvelNNNNNN and surfacexzdispNNNNNN contain respectively on five columns x coordinate in km, z coordinate in km and EW, NS and UD velocities or displacements on the plane y = j0 * ds.
    - o surfaceyzvelNNNNNN and surfaceyzdispNNNNNN contain respectively on five columns y coordinate in km, z coordinate in km and EW, NS and UD velocities or displacements on the plane x = i0 * ds.
- the main frequency of the source function <fd0>. This parameter is used for the CPML.
- the directory were output files are saved <dir>
- the file were sources coordinates are given <fsrcMap>
- the source history file <fsrcHist>
- the file were stations coordinate are given <fstatMap>
- the geological model file <fgeo>
- the coordinates of the geological model origin (lower southwest corner) <x0>, <y0> and <z0>
- the number of geological layers <nlayer>
- for each layer, between the xml markers <layerX> and </layerX>:
    - o the density <rho>
    - o the P-wave velocity <vp>
    - o the S-wave velocity <vs>
    - o the depth of the roof in kilometres <depth>. This value is used only if the option typeModel model = LAYER is chosen.
    - o the name of the layer <name>. This value is used only if the option typeModel model = GEOLOGICAL is chosen.
    - o the quality factor <q0>. This value is used with the option typeAnelas ANLmethod = ANOTHER.
    - o the P-wave quality factor <Qp>. This value is used with the option typeAnelas ANLmethod = DAYandBRADLEY or KRISTEKandMOCZO.
    - o the S-wave quality factor <Qs>. This value is used with the option typeAnelas ANLmethod = DAYandBRADLEY or KRISTEKandMOCZO.
- the parameters for the anelastic method of Day and Bradley (2001): <taum>, <tauM> and <w0>. Day and Bradley proposed to take taum = ω^{-1}, where ω is the Nyquist frequency associated with the time step and tauM = 5 Nω^{-1}, where N is the number of

time steps. The last parameter w0 is the frequency around which quality factors Qp and Qs are computed.
- the parameters for the anelastic method of Kristek and Moczo (2003): <wmin> and <wmax>. These parameters are the boundaries of the interval where quality factors Qp and Qs are computed.

3. Source coordinates file name.map

This file contains on each line the following parameters:
- the number of sources.
- the coordinates along x, y and z of the hypocentre. These data are not used.
- for each source, the number of the source, its coordinate along x, its coordinate along y and its coordinate along z.

4. Source history file name.hist

This file is used only with the option source = HISTFILE.

It contains on each line the following parameters:
- the value of the seismic moment of the source (not used by the code).
- the corresponding value of m_w (not used by the code).
- the value of dsbiem (width of the surface where each source is applied, generally 1) and the value of the time step of the source function.
- the number of time steps where the source function is described.
- for each source:
  - the number of the source and the values of strike, dip and rake (in degrees).
  - the values of the source function at each time step (m').

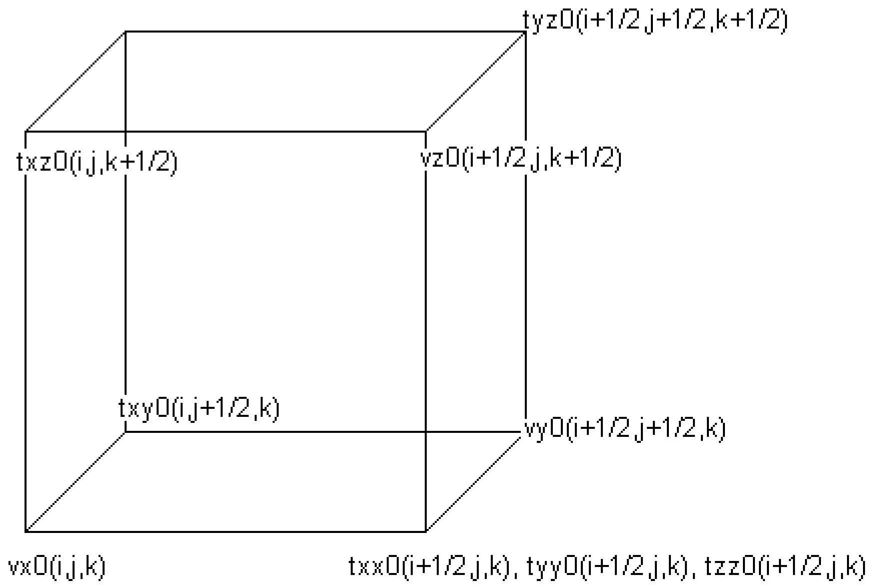
5. Receivers file name_receivers.map

This file contains on each line the following parameters:
- number of receivers.
- with the option model = LAYER: for each receiver, the number of the receiver and its coordinates along x, y and z (in meters).
- with the option model = GEOLOGICAL: for each receiver, the number of the receiver and its coordinate along x and y. The z coordinate is computed by the code.

II. Structure of the code

A. Discretization grid

The discretization grid used in the code is the following:



B. Structure of the code

The structure of the code is the following:

Initialize topology MPI – Call function ReadTopo

↓

Read parameters file – Call function ReadPrmFile

Find string in input file *name.prm* – Call functions File2str, FindDble, FindInt, FindField

↓

Read sources file – Call function ReadSrc

Check if sources are inside the regular domain – Call function WhereAmI

↓

Read stations file – Call function ReadStation

↓

Partition of the domain – Call function InitPartDomain

↓

```
Initialize communications – Call function InitializeCOMM

                                   ↓

If model = GEOLOGICAL: Read geological model file – Call function ReadGeoFile

                                   ↓

Allocate fields – Call function AllocateFields

                                   ↓

Initialize fields

    If model = LAYER: Initialize layer model – Call function InitLayer Model

    If ANLmethod = DAYandBRADLEY: Initialize fields for Day and Bradley (2001)
anelastic method – Call function InitializeDayBradley

    If ANLmethod = KRISTEKandMOCZO: Initialize fields for Kristek and Moczo
(2003) anelastic method – Call function InitializeKristekMoczo

    Initialize fields for absorbing layers – Call function InitializeABC

    If model = GEOLOGICAL
        Write geological model in output file – Call function OutGeol
        Compute the heights of the stations – Call function InitializeGeol

    Initialize outputs – Call function InitializeOutputs

    Print information on medium – Call function PrintInfoMedium

                                   ↓

If PERSISTANT: Prepare sent and received messages between the CPUs

                                   ↓

Beginning of the loop

                                   ↓

    If source = HISTFILE: Increment the seismic moment – Call function
    computeSeisMoment

    ↓

    For imode (compute border cells of each CPU domain before inside cells)
```

Loop for the computation of intermediate values (CMPL/PML), anelastic methods of Kristek & Moczo and Day & Bradley (computeIntermediates.c/h)

↓

```
For i
    For j
        For k
            Computing the position
            If inside the domain or absorbing layer:
                If absorbing layer: computation of the phiV (CPML) or
                t_ijk (PML)
                If anelasticity = Day & Bradley: computation f_ij and
                ksi_ij in the inner domain and the CPML
                If anelasticity = Kristek & Moczo: computation ksil_ij in
                the inner domain and the CPML
            End inside domain or absorbing layer

            If free surface (only xx, yy, xy are computed)
                If k = 1 and anelasticity = Day & Bradley: computation
                If k = 1 and anelasticity = Kristek & Moczo:
                computation of ksil (including ksil_zz)
            End free surface
        End k
    End j
End i
```

Loop for the computation of the stress txx0, tyy0, tzz0, txy0, txz0, tyz0 (computeStress.c)

If PERSISTANT or NONBLOCKING: communication between the CPUs

↓

```
For i
    For j
        For k
            Computing the position
            Computing the material coefficients on the corners of the cube

            If inside the domain or absorbing layer:
                Computing the common part (elastic)
                If absorbing layer: adding the phiV (CPML) or the t_ijk
                (PML)
            End inside domain or absorbing layer

            If free surface
                Computing the common part
                If absorbing layer: adding the phiV (CPML) or the t_ijk
                (PML)
            End free surface
            If anelasticity = ANOTHER: attenuation of the stress
            If anelasticity = Day & Bradley:
                If inner domain: adding ksi_ij
                If free surface: adding ksi_ij (respecting the symmetries)
            If anelasticity = Kristek & Moczo:
                If inner domain: adding ksil_ij
                If free surface: adding ksil_ij
        End k
    End j
End i
```

↓

Communication entre les CPUs

↓

Loop for the computation of the velocity vx0, vy0 et vz0 (computeVeloAndSource.c)

Si PERSISTANT ou NONBLOCKING : communication entre les CPUs

↓

```
For i
    For j
        For k
            Computing the position
            Computing the material coefficients on the corners of the cube

            If inside the domain: computing the velocity
            If free surface: computing the velocity

            If inside the domain and absorbing layer: computing and adding
        the absorbing layers
            If free surface and absorbing layer: computing and adding the
        absorbing layers

            if anelasticity = ANOTHER: attenuation of the velocity
            If source = HISTFILE: adding the source
        End k
    End j
End i
```

↓

If source = VELO: adding the source to the velocity vector

↓

Putting the velocities on the boundaries at zero

↓

Communication between the CPUs

↓

If (time modulo STATION_STEP = 0) computation of the seismograms + communications + writing

↓

If (time modulo SURFACE_STEP = 0) communications + writing of the files surfacexyNNNNNN, surfacexzNNNNNN and surfaceyzNNNNNN

↓

End of the loop

↓

End of the program

C. Functions

Name of file	Name of function	Action
alloAndInit	InitPartDomain	Initialize domain partition
	AllocateFields	Allocate fields
	InitializeCOMM	Initialize communication buffers
	InitializeABC	Initialize (C)PML parameters
	CompABCCoeff	Compute (C)PML parameters
	InitializeGeol	Compute heights of the stations
	InitializeDayBradley	Initialize anelastic parameters for Day and Bradley (2001) method
	InitializeKristek Moczo	Initialize anelastic parameters for Kristek and Moczo (2003) method
	CompYlkap	Compute anelastic coefficients
	inv	Inverse matrix
	InitializeOutputs	Initialize outputs
	DesallocateAll	Desallocate fields
alloAndInit_LayerModel	Initk2ly	Correspondence between depth and layer number
	PrintLayer	Printing information about layer
	GetkBkE	Compute layer top and bottom
	ExtractAlayer	Find layer parameters

	AppendLayer2MDM	Reallocate layer
	AddPart2Layer	Adding part layer to current layer
	EmptyLayer	Put layer parameters to 0
	InitLayerModel	Initialize layer model
computeIntermediates	Q2A	
	LocalComputeDayBradley	
	ComputeIntermediates	
computeStress	ComputeStress	
	ChooseY	
computeVeloAndSource	computeSeisMoment	
	computeVelocity	
inlineFunctions	RhoMuKap2Vp	Compute P-wave velocity from density and shear and bulk moduli
	RhoMu2Vs	Compute S-wave velocity from density and shear modulus
	RhoVpVs2MuKap	Compute shear and bulk moduli from density and P- and S-wave velocities
	WhereAmI	Search where a grid point is inside the computational domain
	IndexL	
	averageInverseDouble8	
	averageInverseDouble4	
	averageInverseDouble2	
	CornerXYZ_GeolInverse	
	CornerXYZ_Geol	
	staggardv4	
	staggards4	
	staggardt4	
	radxx	
	radyy	
	radzz	
	radxy	
	radyz	
	radxz	
	PMLdump4	
	FKSIs4	
	FKSIt4	
	FKSIs4CPML	
	FKSIt4CPML	
	Diff4	
	SumYlKsil	
	test_WhereAmI	
IO	ReadTopo	Read input file topologie.in which contains the number of CPUs along x and y direction
	ReadPrmFile	Read input file xxxx.prm which contains the model parameters
	ReadSrc	Read input file xxxx.map which contains sources coordinates and

		input file xxxx.hist which contains sources history
	ReadStation	Read input file stations.map which contains stations coordinates
	ReadGeoFile	Read file xxxx.vox which contains geological model
	OutSeismograms	
	OutGeol	
	PrintInfoMedium	
	FindField	Find a parameter in parameters file xxxx.prm
	FindInt	Find an integer in parameters file xxxx.prm
	FindDble	Find a double in parameters file xxxx.prm
	File2str	Open parameters file xxxx.prm
main	VerifFunction	Check status of function
	my_second	
	SyncBufStress	
	SyncBufVelocity	
	SyncBufKsil	
	ComputeSeismograms	
	Weight3d	

D. Compilation options

The following options are included in the makefile:

- OUT_HOGE: to write geological model in output files hogeNNN

III.References

Day, S.M. (1998) – Efficient simulation of constant Q using coarse-grained memory variables. *Bull. Seism. Soc. Am.*, **88 (4)**, 1051-1062.

Day, S.M., Bradley, C.R. (2001) - Memory-efficient simulation of anelastic wave propagation. *Bull. Seism. Soc. Am.*, **91**, 529-531.

Emmerich, H., Korn, M. (1987) – Incorporation of attenuation into time-domain computations of seismic wave fields. *Geophysics*, **52 (9)**, 1252-1264.

Graves, R. W. (1996) – Simulating seismic wave propagation in 3D elastic media using staggered-grid finite-differences. *Bull. Seism. Soc. Am.*, **86 (4)**, 1091-1106.

Komatitsch, D., Martin, R. (2007) - An unsplit convolutional Perfectly Matched Layer improved at grazing incidence for the seismic wave equation. *Geophysics*, **72 (5)**, SM155-SM167.

Kristek, J., Moczo, P. (2003) - Seismic-wave propagation in viscoelastic media with material discontinuities: a 3D fourth-order staggered-grid finite-difference modeling. *Bull. Seism. Soc. Am.*, **93**, 2273-2280.

Moczo, P., Kristek, J., Halada, L. (2000) – 3D fourth-order staggered-grid finite-difference schemes: stability and grid dispersion. *Bull. Seism. Soc. Am.*, **90 (3)**, 587-603.

Moczo, P., Kristek, J., Vavryčuk, V., Archuleta, R. J., Halada, L. (2002) – 3D heterogeneous staggered-grid finite-difference modeling of seismic motion with volume harmonic and arithmetic averaging of elastic moduli and densities. *Bull. Seism. Soc. Am.*, **92 (8)**, 3042-3066.