SEM2DPACK

A Spectral Element Method tool for 2D wave propagation
and earthquake source dynamics

User's Guide

Version 2.2.11
June 2007

Jean-Paul Ampuero

ETH Zürich (Swiss Federal Institute of Technology)
Institute of Geophysics
Seismology and Geodynamics Group
ETH Hönggerberg HPP O 13.1
CH-8093 Zürich
Switzerland

E-mail: ampuero@erdw.ethz.ch
Web: www.sg.geophys.ethz.ch/geodynamics/ampuero/
Phone: +41 44 633 2197
Fax: +41 44 633 1065

Chapter 1

Introduction

1.1 Overview

The SEM2DPACK package is a set of software tools for the simulation and analysis of the seismic response of sedimentary basins and dynamic earthquake ruptures. Its core is SEM2D, an explicit Spectral Element solver for the 2D elastic wave equation. The general flow of a simulation project is:

1. Mesh generation: a domain decomposition made of quadrilateral elements, linearly (Q4) or quadratically (Q9) deformed.
2. Mesh quality verification, return to previous step if needed.
3. Numerical simulation.
4. Post-processing, analysis and visualization of the output.

SEM2DPACK provides tools for each step. However, no general mesh generation code is included. Instead SEM2D can import unstructured quadrilateral meshes generated externally. As an example we provide an interface to EMC2, one of the few public domain 2D mesh generators including quadrilateral elements and a Graphical User Interface [1].

This User's Guide explains the usage of the SEM2D solver. For more details, additional results, and for the 3D extension of the algorithm, please refer to Komatitsch (1997), Komatitsch and Vilotte (1998), Komatitsch *et al.* (1999) and also Vai *et al.* (1998).

An introduction to mesh generation with EMC2 is also provided. For more details you must refer to the EMC2 documentation.

This is a research code, constantly under development and provided "as is", and therefore it should *not* be considered by the user as a 100 % bug-free software package. We welcome comments, suggestions, feature requests, module contributions and bug reports.

[1] EMC2 can be downloaded from `http://www-rocq.inria.fr/gamma/cdrom/www/emc2/eng.htm`

1.2 History and credits

The main part of the elastic-isotropic solver was written in the mid 90's by Dimitri Komatitsch while he was preparing his Ph.D. at the *Institut de Physique du Globe de Paris*, under the advise of Prof. Jean-Pierre Vilotte. The elastic-anisotropic solver and several significant improvements to the isotropic code were added by D. Komatitsch later as part of a research contract with DIA Consultants. Further functionalities were added by myself, Jean-Paul Ampuero, while preparing my Ph.D. at IPGP, also under the advise of Prof. Jean-Pierre Vilotte. Most of these additional features were motivated by an ECOS-NORD/FONACYT research project for the study of the seismic response of the valley of Caracas, Venezuela. That became the version 1.0 of the SEM2DPACK, released in April 2002.

For the current version, 2.x, the code was almost completely rewritten in a more object-oriented style in preparation to the implementation of higher level functionalities, such as multigrid, subcycling, adaptivity and multiscale coupling. While the extensive use of object-oriented features of FORTRAN 90 can degrade performance this is not critical in 2D simulation, the emphasis has been rather in facilitating code reuse and expansion.

A simultaneous development for the simulation of earthquake dynamics was undertaken and is the main new feature of the current version. Spontaneous rupture along multiple non-planar faults can be currently modelled. Although there is no intrinsic limitation on applying different friction laws, as of Version 2.2 only linear slip weakening friction is implemented. Dynamic source simulations using methods that discretize the bulk, such as finite difference, finite element and spectral element methods, are more prone to high frequency numerical noise than boundary element methods (e.g. when the size of the process zone is not well resolved). Methods to control this problem were presented in the author's Ph.D. dissertation (Ampuero, 2002)[2] and in Gaetano Festa's Ph.D. dissertation[3], and will be implemented in a forthcoming version of SEM2DPACK.

1.3 Installation

- Uncompress and expand the SEM2DPACK package: `tar xvfz sem2dpack.tgz`
- Go to the source directory: `cd SEM2DPACK/SRC`
- Edit the `Makefile` according to your FORTRAN 95 compiler, following the instructions therein.
- Modify the optimization parameters declared and described in `SRC/constant.f90`.
- Compile: `make`
- Move to the `SEM2DPACK/POST` directory, edit the `Makefile` and compile.

On normal termination you should end up with a set of executable files, among which `sem2dsolve`, in `/home/yourhome/bin/`. I have been developping the code with the Lahey/Fujitsu `lf95` compiler and, more recently, with the Intel compiler for Linux [4]. Other compilers are not being tested on a regular basis, so please report any related problems.

[2] Available in French at `http://www.sg.geophys.ethz.ch/geodynamics/ampuero/phd.html`

[3] `http://people.na.infn.it/~festa/`

[4] This code works properly with the Intel compiler starting with version 8.0.046_pe047.1, so make sure you have a recent version of `ifort` !

1.4 Help requests, feature requests and bug reports

Since November 2006 (version 2.2.5) current and old versions of SEM2DPACK are hosted by SourceForge at `http://sourceforge.net/projects/sem2d/`. To take advantage of the convenient features offered by this host you must create a SourceForge.net account at `http://sourceforge.net/account/newuser_emailverify.php`.

The code repository is at `http://sourceforge.net/project/showfiles.php?group_id=182742`. To receive notification emails about new releases of SEM2DPACK sign up for the "Package Monitor" at `http://sourceforge.net/project/filemodule_monitor.php?filemodule_id=212397`.

A "tracking system" is available at `http://sourceforge.net/tracker/?group_id=182742`, with three separate lists. Requests for implementation of new features must be submitted to the "Feature Requests" tracker. Questions related to the usage of SEM2DPACK must be submitted to the "Support Requests" tracker. Bug reports must be submitted to the "Bugs" tracker. The three tracker lists are browsable and searchable. To browse the complete list of a tracker set "Status" to "Any". Before submitting an issue make sure you are running the most recent version of SEM2DPACK, that you understand the changes listed in SEM2DPACK's `ChangeLog` file and that your problem has not been treated in previous submissions. When relevant, a new submission must include the input files needed to reproduce your problem (`Par.inp`, `*.ftq`, etc). You will receive email notifications of any update of your submitted item, until it is closed. If the item is declared "Pending" you are expected to reply to the last message of the developer within two weeks, otherwise the item will be closed. For more instructions see `http://sourceforge.net/support/getsupport.php?group_id=182742`.

Contributions to SEM2DPACK by experienced programmers are always welcome and encouraged. A "Developers Forum" is available at `http://sourceforge.net/forum/forum.php?forum_id=635737`, where the implementation of new features can be discussed. Although the code is stable for my research purposes, there is still a number of missing features. Their implementation could make SEM2DPACK interesting for a broader audience in mechanical engineering, geotechnical engineering, applied geophysics and beyond. The `ToDo` file included with SEM2DPACK contains a list of missing features that range from basic functionalities to complex code re-engineering.

1.5 License

Chapter 2

The solver SEM2D

2.1 About the method

Based on a mesh of quadrangular spectral elements and a set of material properties, sources and receivers SEM2D solves the elastic wave equation applying a Spectral Element Method (SEM) in space and a second-order explicit Newmark scheme in time. The SEM, introduced by Patera (1984) in Computational Fluid Dynamics, can be seen as a domain decomposition version of Pseudospectral Methods or as a high order version of the Finite Element Method. It inherits from its parent methods the accuracy (spectral convergence) and the geometrical flexibility and natural implementation of mixed boundary conditions, respectively.

Introductory texts to the SEM can be found at `www.math.lsa.umich.edu/~karni/m501/boyd.pdf` (chapter draft, by J.P. Boyd), at `www.mate.tue.nl/people/vosse/docs/vosse96b.pdf` (a tutorial exposition of the SEM and its connection to other methods, by F.N. van de Vosse and P.D. Minev) and at `www.siam.org/siamnews/01-04/spectral.pdf` (a perspective paper). Details about the elastodynamic algorithm and study of some of its properties are presented by Komatitsch (1997), Komatitsch and Vilotte (1998), Komatitsch *et al.* (1999), Komatitsch and Tromp (1999) and Vai *et al.* (1998).

The implementation of fault dynamics is similar to that in FEM, or the "traction at split nodes" method explained by Andrews (1999). More details can be found in the author's Ph.D. dissertation (Ampuero, 2002)[1] and in Gaetano Festa's Ph.D. dissertation[2].

More accesible tutorial code, written in Matlab, can be downloaded from the author's website, at `www.sg.geophys.ethz.ch/geodynamics/ampuero/SEM_matlab.tar.gz`.

2.2 What SEM2D is and what it is not

SEM2D is an explicit spectral element solver for the 2D elastic wave equation. It is written in FORTRAN 90, with some (useful but not essential) FORTRAN 95 features.

There is no built-in general mesh generator for unstructured grids, only some basic mesh functionalities. If your geological model is complicated you need to generate a mesh with some external tool.

[1] `www.sg.geophys.ethz.ch/geodynamics/ampuero/phd.html`
[2] `people.na.infn.it/~festa/`

An example using EMC2 is described in a later chapter.

There is currently a limited number of post-processing and graphic tools included. Most output is in the form of raw binary or ASCII data files and sample scripts are provided for Seismic Unix, Gnuplot and Matlab.

2.3 Format of the input file

The input file must be called `Par.inp`. Most of the file is made of FORTRAN 90's NAMELIST input blocks. The general syntax of a NAMELIST can be found in any FORTRAN 90 textbook and will not be repeated here. The typical structure of `Par.inp` is illustrated by two examples in Figure 2.1 and Figure 2.2. The full documentation of the input blocks is presented after these examples. You should get acquainted with the syntax of the input blocks you are most likely to use. The mandatory or more important input blocks are:

- ECHO
- GENERAL
- MESH_DEF, followed by a MESH_Method block
- MATERIAL
- BC_DEF (one for each boundary condition), followed by a BC_Kind block
- TIME
- SRC_DEF, followed by SRC_TimeFunction and SRC_Mechanism blocks if needed
- REC_LINE

Printed by Jean Paul Ampuero

Jun 01, 07 0:53 **Par.inp** Page 1/1

```
# Parameter file for SEM2DPACK 2.0

#----- Some general parameters ----------------
&ECHO title = 'Test SH', verbose='1111'
    , ItInfo = 1000, ItSnapshots = 100000 /
&GENERAL iexec=1, ngll= 6, fmax=1.25d0 , ndof=1 /

#----- Build the mesh ---------------------------
&MESH_DEF  method = 'CARTESIAN' /
&MESH_CART xlim=0.d0,30.d0 ,zlim=0.d0,30.d0 , nelem=60,60/

#---- Elastic material parameters --------------
&MATERIAL tag=1, mode='ISOTR' /
1.d0 1.7321d0 1.d0

#----- Boundary conditions ---------------------

&BC_DEF  tag = 2 , kind = 'ABSORB' /
&BC_ABSORB side = 'R' , stacey=F/

&BC_DEF  tag = 3 , kind = 'ABSORB' /
&BC_ABSORB side = 'U' , stacey=F/

#---- Time scheme settings ----------------------
&TIME  TotalTime=35.d0, courant = 0.3d0 /

#---- Sources ----------------------------------
&SRC_DEF TimeFunction= 'RICKER' ,coord= 0.d0,0.d0 , mechanism= 'FORCE' /
&SRC_RICKER  f0= 0.5d0, onset = 3.d0, ampli = 0.25d0 /

&SRC_FORCE angle = 0d0/

#----- Receivers ---------------------------------
&REC_LINE number = 7 , field='D', first = 0.d0,0.d0, last = 30d0,0.d0, isamp=1 /


#--------- Plots settings ----------------------
&PLOTS postscript=F , bin=F /
```

Friday June 01, 2007 1/1

Figure 2.1: Input file `Par.inp` for an elementary example in `EXAMPLES/TestSH/` : a boxed region with a structured mesh.

Printed by Jean Paul Ampuero

Jun 01, 07 1:18 **Par.inp** Page 1/1

```
#----- Some general parameters ----------------
&ECHO Title = 'Example 2: using EMC2, Palos Grandes NS' ,
      Verbose='1111', ItInfo = 100, ItSnapshots = 3500 /
&GENERAL Iexec =0 , Ngll = 9 , fmax = 3.0 /
#&GENERAL Iexec =1 , Ngll = 5 , fmax = 3.0 /

#----- Build the mesh ---------------------------
&MESH_DEF  Method = 'EMC2' /
&MESH_EMC2 File= 'NS03qb.ftq'  /

#---- Elastic material parameters --------------

&MATERIAL Tag=1, Mode='ISOTR' /
1800.d0 850.d0 450.d0

&MATERIAL Tag=2, Mode='ISOTR' /
2100.d0 1800.d0 750.d0

&MATERIAL Tag=3, Mode='ISOTR' /
2400.d0 2300.d0 950.d0

&MATERIAL Tag=4, Mode='ISOTR' /
2500.d0 5000.d0 2900.d0

#----- Boundary conditions ---------------------

&BC_DEF  Tag = 2, Kind = 'ABSORB' /
&BC_ABSORB Side='D',Stacey=F /

&BC_DEF  Tag = 3, Kind = 'ABSORB' /
&BC_ABSORB Side='L',Stacey=F /

&BC_DEF  Tag = 4, Kind = 'ABSORB' /
&BC_ABSORB Side='R',Stacey=F /

#---- Time scheme settings ----------------------
&TIME  TotalTime=8.d0, Courant = 0.5d0 /
&TIME_NEWMARK alpha=1.d0, beta=0.d0, gamma=0.5d0 /

#---- Sources ----------------------------------
&SRC_DEF  TimeFunction='RICKER', Mechanism='FORCE', Coord= -1160000.d0,-2000.d0
/
&SRC_RICKER  f0 = 1.d0 , Onset = 1.0d0 ,Ampli = 1.d12 /
&SRC_FORCE Angle = 90. /

#----- Receivers ---------------------------------

# receivers located at the surface by giving a very large vertical position
# locating them at the nearest computational node (AtNode=.true. is the default)
&REC_LINE Number = 11 , First = -1163068.0d0,1.d3, Last = -1159697.36d0,1.d3, Is
amp=130 /

# located inside the medium, not necesarily at a computational node
#&REC_LINE Number = 11 , First = -1163068.0d0,0.d3, Last = -1159697.36d0,0.d3, I
samp=10 , AtNode=.false. /

#--------- Plots settings ----------------------
&PLOTS_POSTSCRIPT  Mesh=F,Vectors=T,Color=T, Interpol = T /
```

Friday June 01, 2007 1/1

Figure 2.2: Input file `Par.inp` for a more realistic example: a sedimentary basin with an unstructured mesh generated by `EMC2`. Available in `EXAMPLES/UsingEMC2/`.

2.3.1 Sources

SRC_DEF

Purpose: Define the sources.

Syntax: `&SRC_DEF TimeFunction,mechanism,coord /`
followed by blocks of the groups SRC_TIMEFUNCTION and SRC_MECHANISM

Arguments:

`TimeFunction`	name	none	The name of the source time function: 'RICKER', 'TAB' or 'STF_USER'
`mechanism`	name	none	The name of the source mechanism: 'FORCE', 'EXPLOSION', 'DOUBLE_COUPLE', 'MOMENT' or 'WAVE'
`coord`	dble	huge	Location of the source (m).
`file`	string	'none'	Station coordinates and delay times can be read from an ASCII file, with 3 columns per line: 1. X position (in m), 2. Z position (in m) and 3. delay (in seconds) that's it.

Notes:

1. bla bla
2. bla bla

Printed by Jean−Paul Ampuero

Aug 16, 07 14:58 **SEM2DPACK input blocks** Page 1/12

```
      =============================================================
      = Self−documentation for the INPUT BLOCKS of the SEM2D code =
      =============================================================



−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−


NAME   : BC_ABSORB
GROUP  : BOUNDARY_CONDITION
PURPOSE: Absorbing boundary
SYNTAX : &BC_ABSORB side,stacey /

 side     [char] [none] Which side of the model corresponds to this
              boundary:       'U'     Up,top
                              'D'     Down,bottom
                              'L'     Left
                              'R'     Right
 stacey   [log] [F] Apply Stacey absorbing conditions for P−SV.
               Presumably higher order than Clayton−Engquist (the default).
 let_wave [log] [T] Allow incident waves across this boundary

NOTE   : Only implemented for vertical and horizontal boundaries.

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−


NAME   : BC_DEF
PURPOSE: Define a boundary condition
SYNTAX : &BC_DEF tag, tags, kind /
           followed eventually by &BC_XXXX blocks

  tag     [int] [none] A number assigned to the boundary. If you are
              using SEM2D built−in structured mesher the conventions are:
                      1       bottom
                      2       right
                      3       up
                      4       left
              If you are importing a mesh, you must use the tags assigned
              to the boundaries during the mesh construction.
  tags    [int(2)] [none] Two tags are needed for interfaces (split−node)
              and for periodic boundaries.
  kind    [char*6] [none] Type of boundary condition. The following are
              implemented:
              'DT0TN0', ' DTTTN0', 'ABSORB', 'PERIOD', 'LISFLT', 'SWFFLT'

NOTE   : you must DEFINE FIRST ALL PERIODIC BOUNDARIES

NOTE   : Some of the boundary conditions need additional data. See their
         respective input blocks if any.

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−


NAME   : BC_LSF
GROUP  : BOUNDARY_CONDITION
PURPOSE: Linear slip fault, a displacement discontinuity interface
         where stress and disp.discont. are linearly related
SYNTAX : &BC_LSF Ktang,Knorm /

 Ktang    [dble] [Inf] Tangential stiffness
 Ctang    [dble] [0d0] Tangential compliance
```

```
 Knorm    [dble] [Inf] Normal stiffness
 Cnorm    [dble] [0d0] Normal compliance

NOTE: for each component you can set K _or_ C, but _not_both_

NOTE: if one of the C=0d0 or K=Inf (the default) then
      no displacement discontinuity is allowed for that component
      (transparent),
      if K=0d0 the fault is a free stress boundary for that component
      In summary the fault can behave as:
              −1      transparent T&N (Tangent and Normal)
              0       stress free T&N
              1       linear−slip/free T, transparent N
              2       transparent T, linear−slip/free N
              3       linear−slip/free T&N

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

NAME   : BC_SWFFLT
GROUP  : BOUNDARY_CONDITION
PURPOSE: Slip weakening friction fault
SYNTAX : &BC_SWFFLT Dc | DcHet, MuS | MuSHet , MuD | MuDHet,
                    Tn | TnHet, Tt | TtHet,
                    Sxx | SxxHet, Sxy | SxyHet, Sxz | SxzHet,
                     Syz | SyzHet, Szz | SzzHet
                    FirstOutput, DtOutput, IxOut /
            followed eventually by distribution input blocks &DIST_XXX
            for Dc,MuS,MuD,Tn and/or Tt (the order is important)

NOTE: for better results, use dynamic faults with the leapfrog time scheme
      and with a layer of damping material (Kelvin−Voigt) near the fault.

Friction law:
 Dc       [dble] [0.5d0] Critical slip
 MuS      [dble] [0.6d0] Static friction coefficient
 MuD      [dble] [0.5d0] Dynamic friction coefficient

Initial stress, can be a superposition of tractions and background stress:
 Tn       [dble] [−100d6] Normal traction (positive = tensile)
 Tt       [dble] [55d6] Tangential traction (positive antiplane: y>0)
 Sxx      [dble] [0d0] sigma_xx
 Sxy      [dble] [0d0] sigma_xy
 Sxz      [dble] [0d0] sigma_xz
 Syz      [dble] [0d0] sigma_yz
 Szz      [dble] [0d0] sigma_zz

NOTE: arguments with the suffix "Het" are used to give
      friction and initial stress parameters non uniform values.
       For instance, DcHet='GAUSSIAN' followed by a DIST_GAUSSIAN block
       sets a gaussian distribution of Dc.
       Several heterogeneous distributions are available,
      See DIST_XXX for their syntax.

For outputs in FltXX_sem2d.dat:
 DtOutput [dble] [0.d0] Time lag between outputs (in seconds)
              Default resets DtOutput = global timestep
 FirstOutput [dble] [0.d0] Start output at this time
 IxOut    [int(3)] [(1,huge,1)] First node, last node and stride
              Default resets Ixout(2)= last point
```

```
NOTE: DtOutput is internally adjusted to the nearest multiple
      of the global timestep

----------------------------------------------------------------------------

NAME   : DIST_GAUSSIAN
GROUP  : DISTRIBUTIONS_2D
PURPOSE: Bell shaped (Gaussian) 2D distribution
SYNTAX : &DIST_GAUSSIAN centered_at, length, offset, ampli /

 centered_at      [dble(2)] [none] Coordinates of the center point.
 length           [dble(2)] [none] Characteristic lengths on each axis.
 offset           [dble] [none]    Background level.
 ampli            [dble] [none]    Amplitude from background.

----------------------------------------------------------------------------

NAME   : DIST_GRADIENT
GROUP  : DISTRIBUTIONS_2D
PURPOSE: Constant gradient 2D distribution.
SYNTAX : &DIST_GRADIENT file,valref ,grad,angle/

 file             [name] [none]    Name of the file containing the coordinates
                       of the points defining the reference line.
                       It is an ASCII file with 2 columns per line:
                       (1) X position (in m) and
                       (2) Z position (in m)
 valref           [dble] [none]    Value along the reference line
 grad             [dble >0] [none] Positive gradient (valref_units/meter)
 angle            [dble] [none]    Angle (degrees) between the vertical down
                       and the grad+ direction. Anticlockwise convention (grad+
                       points down if 0, right if 90)

NOTE   : Be sure that your angle and ref−line are compatible. The code will
         abort if the ref−line is too short: some points of the domain
         cannot be projected to ref−line in the angle direction.

----------------------------------------------------------------------------

NAME   : DIST_HETE1
GROUP  : DISTRIBUTIONS_2D
PURPOSE: Linear interpolation of values from a regular 2D grid.
SYNTAX : &DIST_HETE1 file, col /

 file             [name] [none] Name of the file containing the definition
                      of the regular grid and values at grid points.
                      The format of this ASCII file is:
                         Line 1 :  ncol nx nz x0 z0 dx dz
                           ncol  = [int] number of data columns
                           nx,nz = [2*int] number of nodes along x and z
                           x0,z0 = [2*dble] bottom−left corner
                           dx,dz = [2*dble] spacing along x and z
                         Line 2 to nx*nz+1 : [ncol*dble] values at grid points
                           listed from left to right (x0 to x0+nx*dx),
                           then from bottom to top (z0 to z0+nz*dx)
 col              [int] [1]    Column of the file to be read

NOTE   : The same file can contain values for (ncol) different properties,
```

```
         (e.g. rho, vp, vs) but each DIST_HETE1 block will read only one.

NOTE   : Even if the original model domain has an irregular shape,
         the regular grid where input values are defined must be rectangular
         and large enough to contain the whole model domain.
         The regular grid possibly contains buffer areas with dummy values.
         These dummy values should be assigned carefully (not random nor zero)
         because SEM2D might use them during nearest−neighbor interpolation.

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

NAME   : DIST_LINEAR
GROUP  : DISTRIBUTIONS_1D
PURPOSE: Piecewise linear 1D distribution along X.
SYNTAX : &DIST_LINEAR file,length /

 file     [name] [none] Name of the ASCII file containing
              the data to be interpolated, two columns per line:
              (1) X position, sorted in increasing order, and
              (2) data value at X
 length   [dble] [0]    Smoothing length for sliding average window
                            No smoothing if length=0

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

NAME   : DIST_ORDER0
GROUP  : DISTRIBUTIONS_2D
PURPOSE: Blockwise constant 2D distribution.
SYNTAX : &DIST_ORDER0 xn, zn /
         x(1) ...  x(xn−1)
         z(1) ...  z(zn−1)
         v(1,1)  ... v(xn,1)
           ...    ...   ...
         v(1,zn) ... v(xn,zn)

 xn       [int] [none] Number of zones along X
 zn       [int] [none] Number of zones along Z
 x        [dble(xn−1)] [none] Boundaries of X−zones: first zone X < x(1),
               second zone x(1) < X < x(2), ... , last zone x(xn−1) < X
 z        [dble(zn−1)] [none] Boundaries of Z−zones
 v        [dble(xn,zn)] [none] Values inside each zone

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

NAME   : DIST_PWCONR
GROUP  : DISTRIBUTIONS_2D
PURPOSE: Piecewise constant radial (2D) distribution.
SYNTAX : &DIST_PWCONR num, ref /
            r(1)  ... ...  r(num−1)
         v(1) v(2) ... v(num−1) v(num)

 num      [int] [none] Number of radial zones (including outermost)
 ref      [dble(2)] [(0d0,0d0)] Reference point: center of radial zones
 r        [dble(num−1)] [none] External radius of zones:
               first zone R < r(1), second r(1) =< R < r(2), ...
               last r(num−1) =< R
 v        [dble(num)] [none] Values inside each zone

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
```

```
NAME   : DIST_SPLINE
GROUP  : DISTRIBUTIONS_1D
PURPOSE: Spline interpolated 1D distribution along X.
SYNTAX : &DIST_SPLINE file /

 file     [name] [none] Name of the ASCII file containing
              the data to be interpolated, two columns per line:
              (1) X position, sorted in increasing order, and
              (2) data value at X

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

NAME   : ECHO
PURPOSE: Parameters controlling runtime output
SYNTAX : &ECHO Verbose,ItInfo,ItSnapshots,ItSnapshot1 /

 Title            [word] [none] Title of the simulation
 Verbose          [char(4)] ['1101'] Verbose flags for input,initialization,
                       check and solver phases. Example: '0001' is verbose only
                       during solver.
 ItInfo           [int] [100] Frequency (in number of timesteps) at which
                       solver echoes some basic information.
 ItSnapshots      [int] [100] Frequency (in number of timesteps) at which
                       snapshots are dumped (usually PostScript)
 ItSnaphot1       [int] [0]   Time step at which first snapshot is dumped

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

NAME   : GENERAL
PURPOSE: General parameters
SYNTAX : &GENERAL iexec,ngll,fmax /

 iexec    [int] [0] Run level:
                      0 = just check
                      1 = solve
 ngll     [int] [9] Number of GLL nodes per edge on each spectral element
              ( polynomial order +1 ). Usually 5 to 9.
 fmax     [dble] [0.d0] Maximum frequency to be well resolved. Mandatory.
              This is a target frequency, the code will check if it is
              compatible with the mesh and eventually issue a warning. To
              improve the resolution for a given fmax you must increase ngll
              (but you will have to use shorter timesteps) or refine/redesign
              the mesh.
 ndof     [int] [2] Number of degrees of freedom per node
                       1 = SH waves, anti−plane
                       2 = P−SV waves, in−plane

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

NAME   : MATERIAL
PURPOSE: Define elastic material properties of a tagged domain
SYNTAX : &MATERIAL tag, mode /
         Followed by material data, with format depending on the mode (see
         below)

 tag      [int] [none] Number identifying a mesh domain
 mode     [char*5] ['ISOTR'] Type of material and/or spatial distribution.
              The following modes are implemented and this is their data
```

```
              format:

              'ISOTR' homogeneous isotropic elastic
              One line, dble(3):
              density, P−wave−velocity, S−wave−velocity

              'ANISO' homogeneous anisotropic
              One line, dble(5):
              density, c11, c13, c33, c44

              'XXXXX' isotropic with any 2D distribution
              Three $DIST_XXXXX blocks:
              density, P−velocity, S−velocity

NOTE   : two MATERIAL blocks can share the same domain tag,
         for instance to assign elastic and plastic material properties
         to the same domain

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

NAME   : KELVIN_VOIGT
PURPOSE: Define Kelvin−Voigt viscosity properties (whole domain)
         i.e. add damping term C*v = K*eta*v
         (eta is a viscous time)
SYNTAX : &KELVIN_VOIGT eta|etaH, ETAxDT /

 eta   [dble][0d0] Viscosity coefficient
 etaH  [char*][] If eta is distributed non uniformly
                 give here the nameof the distribution,
                 followed by a DIST_XXX input block.
 ETAxDT        [log][T] If eta is given in units of dt (timestep)

NOTE: useful as artificial damping layer in fault zones to control
       high frequency noise. Set eta=0.1*dt and a thickness of 4−5 GLL nodes.

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

NAME   : MESH_CART
GROUP  : MESH_DEF
PURPOSE: Rectangular box with structured mesh.
SYNTAX : &MESH_CART xlim,zlim,nelem /

 xlim     [dble(2)] [none] X limits of the box (min and max)
 zlim     [dble(2)] [none] Z limits of the box (min and max)
 nelem    [int(2)] [none]  Number of elements along each direction
 FaultX   [log] [F] Cut the box in the middle by a horizontal fault
                        If enabled, nelem(2) must be even

NOTE: the following tags are automatically assigned to the boundaries:
              1       Bottom
              2       Right
              3       Top
              4       Left
              5       Fault, bottom side
              6       Fault, top side

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

NAME   : MESH_CART_DOMAIN
```

```
PURPOSE: Define a subdomain within a structured meshed box.
SYNTAX : &MESH_CART_DOMAIN tag,ex,ez /

 tag       [int] [none] Tag number assigned to this domain.
 ex        [int(2)] [none]  First and last element along the X direction.
 ez        [int(2)] [none]  First and last element along the Z direction.

NOTE   : If you ignore this input block a single domain (tag=1) will span
         the whole box

-----------------------------------------------------------------------------

NAME   : MESH_EMC2
GROUP  : MESH_DEF
PURPOSE: Imports a mesh from INRIA's EMC2 mesh generator in FTQ format
SYNTAX : &MESH_EMC2 file /

 file      [name] [none] Name of the FTQ file, including suffix

-----------------------------------------------------------------------------

NAME   : MESH_DEF
PURPOSE: Selects a method to import/generate a mesh.
SYNTAX : &MESH_DEF method /

 method    [name] [none] 'CARTESIAN', 'LAYERED' or 'EMC2'
               The &MESH_DEF input block must be followed by a
               &MESH_method input block

-----------------------------------------------------------------------------

NAME   : MESH_LAYERED [mesh]
PURPOSE: Structured mesh for layered medium
         with surface and interface topography.
SYNTAX : &MESH_LAYERED xlim,zmin,nx,file,nlayer /

 xlim      [dble(2)] [none] X limits of the box (min and max)
 zmin      [dble] [none] bottom Z limit of the box
 nx        [int] [none]  Number of elements along X direction
 file      [string] [''] Only for flat layers,
                name of ASCII file containing layer parameters,
                one line per layer, listed from top to bottom,
                3 columns per line:
                (1) vertical position of top boundary,
                (2) number of elements along Z direction
                (3) material tag
 nlayer    [int] [none]  Number of layers
                If a file name is not given the layer parameters
                must be given immediately after the &MESH_LAYERED block
                by nlayer &MESH_LAYER input blocks,
                one for each layer, listed from top to bottom.

NOTE: the following tags are automatically assigned to the boundaries:
              1       Bottom
              2       Right
              3       Top
              4       Left
              5       Fault, bottom side
              6       Fault, top side
```

```
-----------------------------------------------------------------------------

NAME   : MESH_LAYER
GROUP  : MESH_DEF
PURPOSE: Define mesh parameters for one layer
SYNTAX : &MESH_LAYER nz, ztop, ztopH, tag /

 nz       [int] [none]  Number of elements in layer along Z direction
 ztop     [dble] [none] Only for layers with flat top surface:
               vertical position of top boundary
 ztopH    [string] ['none'] Only for layers with irregular top boundary:
               name of distribution, 'LINEAR', 'SPLINE' or any other
               1D distribution available through a DIST_XXXX block.
               If ztopH is set, the MESH_LAYER block must be
               followed by the appropriate DIST_XXXX block.
 tag      [int] [none]  Material tag
                 If not given, a tag is automatically assigned to the layer,
                sequentially numbered from top to bottom (top layer tag =1)

-----------------------------------------------------------------------------

NAME   : PLOTS
PURPOSE: Selects a format to export snapshots
SYNTAX : &PLOTS fields,components, bin,visual3,avs,postscript,gmt /

 fields   [char*] ['V'] fields to export in snapshots
                (begining of output file names given in parenthesis)
                'D'     displacements (dx,dy,dz,da)
                'V'     velocity (vx,vy,vz,va)
                'A'     acceleration (ax,ay,az,aa)
                'E'     strain (e11,e22,e12,e23,e13)
                'S'     stress (s11,s22,s12,s33,e13,e23)
 components [char*] ['ya'] components for PostScript outputs
                       'x','y','z' and/or 'a' (amplitude)
                       (in SH only 'y' is considered)
 postscript [log] [T] PostScript
 gmt      [log] [F] output triangulation file grid_sem2d.gmt
                to be used in "pscontour -T" of the General Mapping Tool (GMT)
 avs      [log] [F] AVS
 visual3  [log] [F] Visual3
 bin      [log] [T] binary

NOTE   : If you choose PostScript you may need also a $POSTSCRIPT input block.
         Other formats apply only to 'DVA' fields, 'ES' are exported as binary.

-----------------------------------------------------------------------------

NAME   : PLOTS_POSTCRIPT
GROUP  : PLOTS
PURPOSE: Preferences for PostScript snapshots
SYNTAX : &PLOTS_POSTSCRIPT vectors, mesh, background, color,
              isubsamp, boundaries, symbols, numbers, legend,
              ScaleField, Interpol, DisplayPts /

 vectors          [log] [F] Plots a vectorial field with arrows
 mesh             [log] [F] Plots the mesh on background
 background       [char] [''] Filled background, only for vector plots:
                                ''   none
```

```
                               'P'  P−velocity model
                               'S'  S−velocity model
                               'T'  domains
 isubsamp          [int] [3] Subsampling of the GLL nodes for the
                           output of velocity model.
                           The default samples every 3 GLL points.
 boundaries        [log] [T] Colors every tagged boundary
 symbols           [log] [T] Plots symbols for sources and receivers
 numbers           [log] [F] Plots the element numbers
 legend            [log] [T] Writes legends
 color             [log] [T] Color output
 ScaleField        [dble] [0d0] Fixed amplitude scale (saturation),
                       convenient for comparing snapshots and making movies.
                       The default scales each snapshot by its maximum amplitude
 Interpol          [log] [T] Interpolate field on a regular subgrid
                       inside each element
 DisplayPts        [log] [3] Size of interpolation subgrid inside each
                       element is DisplayPts*DisplayPts. The default plots at
                       vertices, mid−edges and element center.

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

NAME   : REC_LINE
PURPOSE: Defines a line of receivers
SYNTAX : &REC_LINE number,isamp,field,first,last,file,AtNode,irepr /

 number   [int] [0] Number of stations in the line
 isamp    [int] [1] Sampling stride (in number of timesteps). Note that
               for stability reasons the timestep can be very small.
 field    [char] ['V'] The field in the seismogram:
                           'D'     displacement
                           'V'     velocity
                           'A'     acceleration
 first    [dble(2)] Receivers can be located along a line,
               this is the position (x,z) of the first receiver
 last     [dble(2)] Position (x,z) of the last receiver,
               other receivers will be located with regular spacing
               between First and Last.
 file     [name] ['none'] Station positions can instead be read
               from an ASCII file, with 2 columns per line:
               (1) X position (in m) and
               (2) Z position (in m)
 AtNode        [log] [T] Relocate the stations at the nearest GLL node
 irepr    [char] ['D'] Abscissa for the seismic multitrace plot:
                           'X' Horizontal position
                           'Z' Depth
                           'D' Distance to the first station

NOTE   : to locate receivers at the free surface set their vertical position
         above the free surface and AtNode=T

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

NAME   : SRC_FORCE
GROUP  : SRC_MECHANISM
PURPOSE: Point force source
SYNTAX : &SRC_FORCE angle /

 angle    [dble] [0d0] For P−SV, the angle of the applied force,
```

Printed by Jean−Paul Ampuero


```
                 in degrees, counterclockwise from Z−UP, e.g.:
                 (90 points left, 180 points down)
                 For SH, angle is ignored.

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

NAME   : SRC_DEF
PURPOSE: Define the sources.
SYNTAX : &SRC_DEF   TimeFunction,mechanism,coord /
         followed by blocks of the groups SRC_TIMEFUNCTION and SRC_MECHANISM

 TimeFunction  [name] [none] The name of the source time function:
                    'RICKER', 'TAB' or 'STF_USER'
 mechanism     [name] [none] The name of the source mechanism:
                    'FORCE', 'EXPLOSION', 'DOUBLE_COUPLE', 'MOMENT' or 'WAVE'
 coord         [dble] [huge] Location of the source (m).
 file          [string] ['none'] Station coordinates and delay times can
                    be read from an ASCII file, with 3 columns per line:
                    (1) X position (in m),
                    (2) Z position (in m) and
                    (3) delay (in seconds)

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

NAME   : SRC_DOUBLE_COUPLE
GROUP  : SRC_MECHANISM
PURPOSE: Define a double−couple source
SYNTAX : &SRC_DOUBLE_COUPLE  dip /

 dip           [dble] [90] Dip angle, in degrees, clockwise
                   from the positive X direction

NOTE   : Sign convention: if the source amplitude is positive the right block
         moves up (positive Z direction) in PSV and forward (positive Y
         direction) in SH.

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

NAME   : SRC_MOMENT
GROUP  : SRC_MECHANISM
PURPOSE: Define a moment tensor source
SYNTAX : &SRC_MOMENT Mxx,Mxz,Mzx,Mzz , Myx,Myz /

 Mxx,Mxz,Mzx,Mzz [dble] [0] Tensor components for PSV
 Myx,Myz       [dble] [0] Tensor components for SH

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

NAME   : SRC_WAVE
GROUP  : SRC_MECHANISM
PURPOSE: Incident plane wave through the absorbing boundaries
SYNTAX : &SRC_WAVE angle, phase /

 angle    [dble] [0d0]    Incidence angle in degrees within [−180,180]
                counterclockwise from the positive Z (up) direction
                to the wave vector direction:
                Exs: incidence from below if angle in ]−90,90[
                     normal incidence from below if angle=0
                     from bottom right if angle=+45
```

```
                         from bottom left if angle=-45
 phase    [char] ['S']    'S' or 'P' (only needed in PSV, ignored in SH)

NOTE    : Incident waves enter through the absorbing boundaries.
          An incident wave is applied on every absorbing boundary
          unless "let_wave = F" in the respective BC_ABSO block.
          Incident waves are not implemented for "Stacey" absorbing boundaries.

----------------------------------------------------------------------------


NAME    : STF_RICKER
GROUP   : SRC_TIMEFUNCTION
PURPOSE: The Ricker wavelet is the second derivative of a gaussian.
SYNTAX : &STF_RICKER ampli, f0, onset /

 ampli    [real] [1.] Signed amplitude of the central peak
 f0       [real >0] [0] Fundamental frequency (Hz).
                distribution: it has a peak at f0 and an exponential
                decay at high frequency. The cut-off high frequency is usually
                taken as fmax = 2.5 x f0.
 onset    [real >1/f0] [0] Delay time (secs) with respect to the peak value.

NOTE    : The spectrum has a peak at f0 and decays exponentially at high
          frequencies. Beyond 2.5*f0 there is little energy, this is a
          recommended value for fmax.
NOTE    : onset>1/f0 is needed to avoid a strong jump at t=0, which can cause
          numerical oscillations. Ignore if using incident waves.

----------------------------------------------------------------------------


NAME    : STF_TAB
GROUP   : SRC_TIMEFUNCTION
PURPOSE: Source time function spline-interpolated from values in a file
SYNTAX : &STF_TAB file /

 file     [string] ['stf.tab'] ASCII file containing the source time function,
                two columns per line:
                (1) time
                (2) value

NOTE    : time can be irregularly sampled
NOTE    : assumes value=0 before min(time) and after max(time)

----------------------------------------------------------------------------


NAME    : STF_USER
GROUP   : SRC_TIMEFUNCTION
PURPOSE: User-supplied source time function (a template)
SYNTAX : &STF_USER ampli, onset, par1, par2, ipar1, ipar2 /

 ampli    [dble] [1.] Amplitude
 onset    [dble] [0]  Delay time (secs)
 par1     [dble] [0]  Example parameter
 par1     [dble] [0]  Example parameter
 par1     [int] [0]  Example parameter
 par1     [int] [0]  Example parameter

NOTE    : The user must modify the template module stf_user.f90
```

```
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

NAME   : TIME
PURPOSE: Defines time integration scheme
SYNTAX : &TIME kind, NbSteps, Dt, Courant, TotalTime /

 kind     [char*10] ['leapfrog'] Type of scheme:
                         'newmark'       Newmark−alpha
                         'leapfrog'      Central difference
                         'symp_PV'               Position Verlet
                         'symp_PFR'      Position Forest−Ruth (4th order)
                         'symp_PEFRL'    Extended PFR (4th order)
 NbSteps  [int] [none] Number of timesteps to be performed
 Dt       [dble] [none] Amplitude of the timestep
 Courant  [dble] [0.5d0] Courant stability number: the maximum ratio
              Dt*wave_velocity/dx where dx is the inter−GLL node distance
              Tipically <= 0.5
 TotalTime[int] [none] Total duration (in seconds) of simulation

NOTE   :      Not all combinations of parameters need to be set at once.
              You can set the total duration (secs) or the number of steps.
              You can set the timestep or the Courant number (or use default).

NOTE:          The leap−frog scheme is equivalent to the Newmark scheme
               with alpha=1, beta=0, gamma=1/2.
               However it is faster and requires less memory.
               Dynamic faults require this scheme.

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

NAME   : TIME_NEWMARK
PURPOSE: Parameters of the explicit Newmark or HHT−alpha time scheme
SYNTAX : &TIME_NEWMARK alpha|gamma, beta|rho /

 beta     [dble] [0.5d0] The algorithm is fully explicit if beta=0
              otherwise it is a single−predictor−corrector scheme
 gamma    [dble] [1.d0]
 alpha    [dble] [0.5d0] parameter in the Hilber−Hughes−Taylor method
              Actually, here alpha = 1 + their original definition of alpha
 rho      [dble] [1.d0] high frequencies are damped by a factor>=rho.
              The default is non−dissipative. Dissipation is limited however
              to rho>=0.5 . For max dissipation you should work close to
              the stability limit (Courant around 0.56 for rho=0.5).

NOTE: For second order schemes only two parameters need to be set:
      (alpha OR gamma) AND (beta OR rho)

NOTE: Dissipative schemes (0.5<=rho<1) are slightly more unstable,
      i.e. they require slightly smaller Courant number
      (0.56 for rho=0.5, compared to 0.6 for rho=1)
```

2.4 Verifying the settings and running a simulation

Once the code has been successfully compiled, the simulation can be started by typing `sem2dsolve` from your working directory, which contains the file `Par.inp`. The computations can be run in background and the screen output saved in a file (e.g. `info`) by typing `sem2dpack > info &`.

A typical screen output of SEM2D, corresponding to the first example, is shown on the following pages. The parameters of the simulation and some verification information are reported there in a self-explanatory form. You are advised to do a first run with `iexec=0` in the `GENERAL` input block and check all these informations prior to the real simulation. You should always verify the following:

- **Stability:** the CFL stability number should be smaller than $0.55 \sim 0.60$ for second order time schemes. This number is defined at each computational node as

$$\mathrm{CFL} = c_P \ \Delta t/\Delta x$$

  where Δt is the timestep, c_P the P-wave velocity and Δx the local grid spacing. Note that Δx is usually much smaller than the element size h ($\approx$ `Ngll`^{2} times smaller) because SEM internally subdivides each element onto a non-regular grid of `Ngll`×`Ngll` nodes clustered near the element edges (Gauss-Lobatto-Legendre nodes). If the computation is unstable, the maximum displacement, printed every `ItInfo` time steps, increases exponentially with time. Stability can be controlled by decreasing `Dt` or `Courant` in `Par.inp`.

- **Resolution:** the number of nodes per shortest wavelength λ_{min} should be larger than $4.5 \sim 5$. The minimum wavelength is defined as

$$\lambda_{min} = \min(c_S)/f_{max}$$

  where c_S is the S-wave velocity and f_{max} the highest frequency you would like to resolve, e.g. the maximum frequency at which the source spectrum has significant power (for a Ricker wavelet $f_{max} = 2.5 \times f_0$). For an element of size h and polynomial order $p =$ `Ngll` $- 1$, the number of nodes per wavelength G is

$$G = \frac{p\,\lambda_{min}}{h}.$$

  Typical symptoms of poor resolution are ringing and dispersion of the higher frequencies. However, in heterogeneous media these spurious effects might be hard to distinguish from a physically complex wavefield, so mesh resolution must be checked beforehand. If resolution is too low the mesh might be refined by increasing `Ngll` in `Par.inp` (p-refinement) or by generating a denser mesh (h-refinement). If you were using EMC2 as a mesh generator, the script `PRE/href.csh` can be useful for h-refinement.

- **Cost:** the total CPU time an memory required for the simulation are as much as you can afford. Estimates of total CPU time are printed at the end of check mode. Details about memory usage can be found in `MemoryInfo_sem2d.txt`.

The quality of the mesh can be inspected with the Matlab script `PRE/ViewMeshQuality.m` which produces plots like in Figure 2.3. The proper balance of the mesh with respect to the following two criteria can be analyzed:

- **Stability criterion**, related to the largest stable timestep. On each element we define a stability index as the logarithm of $\min(\Delta x/c_P)$ normalized by its median value over the whole

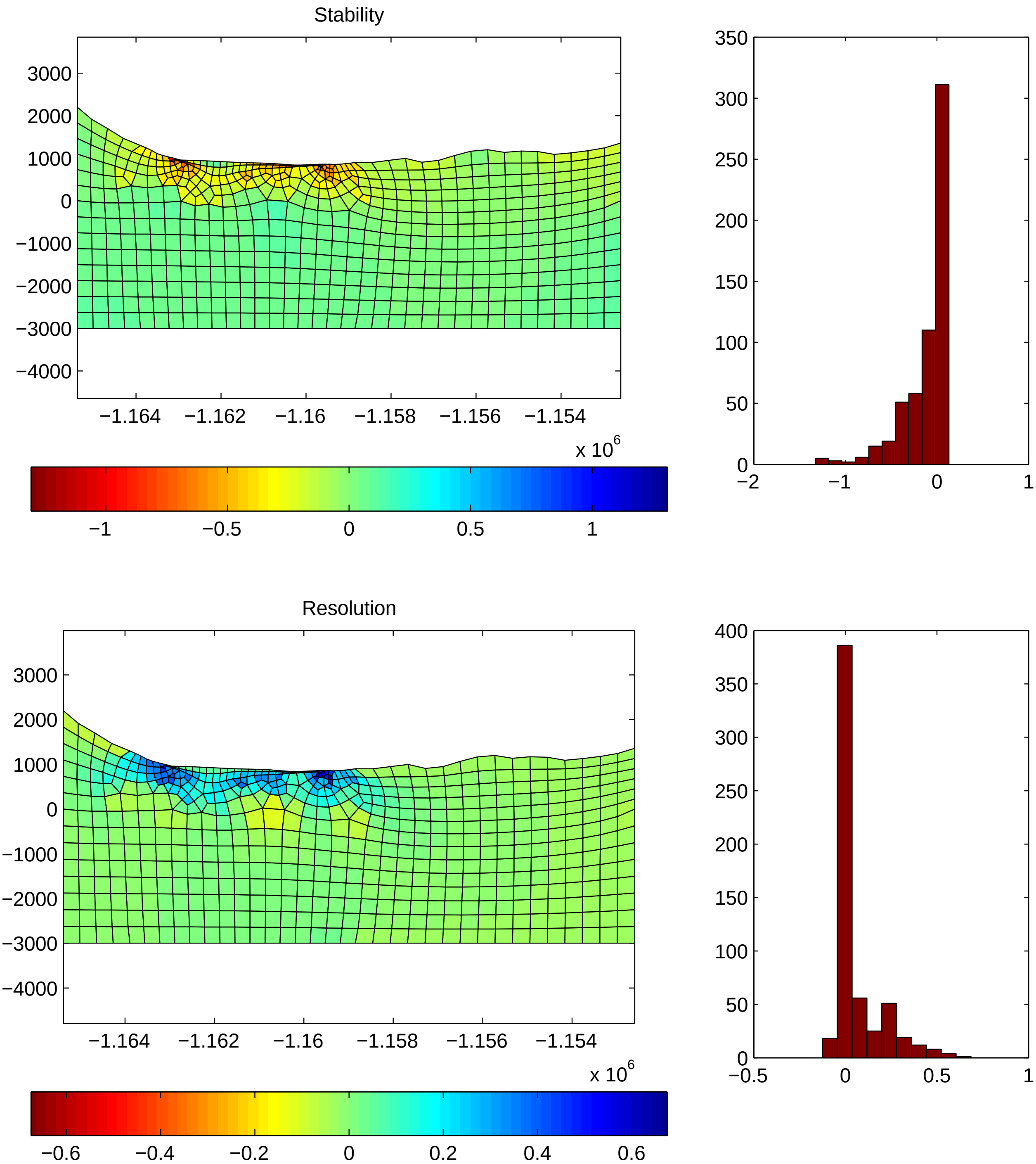

Figure 2.3: Checking the quality of a mesh with `PRE/ViewMeshQuality.m` for the example in `EXAMPLES/UsingEMC2/`. The balance of the stability and resolution properties of the mesh can be analyzed: logarithmic stability index (top) and logarithmic resolution index (bottom). Histograms of these indices (in number of elements) are shown on the right.

mesh. Red elements (small stability index) are relatively unstable and require small timesteps Δt. Because Δt is constant over the whole mesh and the computational cost is inversely proportional to Δt these red elements penalize the computational efficiency. The mesh should be redesigned to increase their size, as much as possible, while keeping them small enough to resolve the shortest wavelength (see next).

- **Resolution criterion**, related to the number of nodes per shortest wavelength. On each element we define a resolution index as the logarithm of $\min(c_S/h)$ normalized by its median value over the whole mesh. Red elements (small resolution index) have relatively poor resolution, in their vicinity the maximum frequency resolvable by the mesh is limited. The mesh should be redesigned to decrease their size, as much as possible. Conversely, elements with very high resolution index (blue) are smaller than required and might increase the computational cost.

To minimize the CPU and memory cost of a simulation an ideal mesh design should minimize the spread of the two indices above, by aiming at a ratio of element size to wave velocity, h/c, as uniform as possible across the whole mesh. However, in some cases a poorly balanced mesh is inevitable: in the example of Figure 2.3 the worst elements are near the edges of the sedimentary basin, at a sharp velocity contrast. Small element sizes on the rock side are inherited from the sediment mesh.[3]

Similar information is plotted by `gv Stability_sem2d.ps` and `gv Resolution_sem2d.ps`. The indices in these files are however not logarithmic and are not normalized by the median.

[3] In future releases of SEM2DPACK this penalty on computational efficiency will be reduced by non-conformal meshing with mortar elements, by timestep subcycling or by implicit/explicit timestep partitioning.

```
 -------------------------------------------------------------------------------
 -------------------------------------------------------------------------------
 Program  S P E C F E M : start
 -------------------------------------------------------------------------------
 -------------------------------------------------------------------------------

 -------------------------------------------------------------------------------
 -------------------------------------------------------------------------------
 D a t e : 01 - 06 - 2007                                 T i m e  : 01:06:00
 -------------------------------------------------------------------------------
 -------------------------------------------------------------------------------

***********************************************
*          I n p u t   p h a s e              *
***********************************************



 E c h o   S e t t i n g s
 ========================


     Echo during input phase. . . . . . . . . . . . (Verbose(1)) = T
     Echo during init phase . . . . . . . . . . . . (Verbose(2)) = T
     Echo during check phase. . . . . . . . . . . . (Verbose(3)) = T
     Echo during run phase. . . . . . . . . . . . . (Verbose(4)) = T
     Display frequency  . . . . . . . . . . . . . .(ItSnapshots) = 100000
     First display. . . . . . . . . . . . . . . . .(ItSnapshot1) = 0
     Basic info output frequency. . . . . . . . . . . (ItInfo) = 1000



 G e n e r a l   P a r a m e t e r s
 ===================================


     Execution mode . . . . . . . . . . . . . . . . (iexec) = solve
     Number of nodes per edge . . . . . . . . . . . .(ngll) = 6
     Number of d.o.f per node . . . . . . . . . . . .(ndof) = 1
     Highest frequency to be resolved . . . . . . . .(fmax) =    1.250E+00


 M e s h   G e n e r a t i o n
 =============================


     Method. . . . . . . . . . . . . . . . . . .(method) = CARTESIAN
     Minimum X . . . . . . . . . . . . . . . (xlim(1)) =  0.000E+00
     Maximum X . . . . . . . . . . . . . . . (xlim(2)) =  3.000E+01
     Minimum Z . . . . . . . . . . . . . . . (zlim(1)) =  0.000E+00
     Maximum Z . . . . . . . . . . . . . . . (zlim(2)) =  3.000E+01
     Number of elements along X. . . . . .(nelem(1)) =    60
     Number of elements along Z. . . . . .(nelem(2)) =    60
     Cut by horizontal fault . . . . . . . .(faultx) = F


 T i m e   i n t e g r a t i o n
 ==============================


     Scheme. . . . . . . . . . . . . . .(kind) = leapfrog
     The number of steps will be set later
     The timestep will be set later
     Courant number. . . . . . . . .(Courant) = 0.30
     Total simulation duration . (TotalTime) =   35.000E+00


 M a t e r i a l   s e t s :   2 D  e l a s t i c i t y
 ======================================================


     Number of material sets . . . . . . . . . =     1
```



```
     Material number . . . . . . . . . . . .(tag) =     1
     Type  . . . . . . . . . . . . . . . . .(mode) = ISOTR
     P-wave velocity . . . . . . . . . . . .(cp) =   1.732E+00
     S-wave velocity . . . . . . . . . . . .(cs) =   1.000E+00
     Mass density. . . . . . . . . . . . (denst) =   1.000E+00
     Poisson's ratio . . . . . . . . . . (poiss) = 250.021E-03
     First Lame parameter Lambda . . . .(alam) =   1.000E+00
     Second Lame parameter Mu. . . . . . (amu) =   1.000E+00
     Bulk modulus K. . . . . . . . . . . .(Kvol) =   1.667E+00
     Young's modulus E. . . . . . . . (young) =   2.500E+00


 B o u n d a r y   C o n d i t i o n s
 =====================================



     Boundary tag. . . . . . . . . . . . . (tag) = 2
     Boundary condition. . . . . . . . . .(kind) = ABSORB
     Type of absorbing boundary. . . .(stacey) = Clayton-Engquist
     Periodicity . . . . . . . . . .(periodic) = F

     Boundary tag. . . . . . . . . . . . . (tag) = 3
     Boundary condition. . . . . . . . . .(kind) = ABSORB
     Type of absorbing boundary. . . .(stacey) = Clayton-Engquist
     Periodicity . . . . . . . . . .(periodic) = F


 S o u r c e   F u n c t i o n s
 ==============================


     X-position (meters). . . . . . . . . . =    0.000E+00
     Y-position (meters). . . . . . . . . . =    0.000E+00
     Function Type. . . . . . . . . . . . . . = Ricker
     Fundamental frequency (Hz) . . . . . . . = 500.000E-03
     Time delay (s) . . . . . . . . . . . . =   3.000E+00
     Multiplying factor . . . . . . . . . . . = 250.000E-03
     Source Type. . . . . . . . . . . . . . . = Collocated Force
     If P-SV: counterclockwise angle / up . = 0.00


 R e c e i v e r s
 =================


     Number of receivers . . . . . . . . . . . . . (number) = 7
     Subsampling for seismograms recording . . . .(isamp) = 1
     Field recorded. . . . . . . . . . . . . . . . . .(field) = D
     Axis of the seismogram plot . . . . . . . . . .(irepr) = D


 S n a p s h o t   O u t p u t s
 ===============================


     Save results in PS file or not . . . . .(postscript) = F
     Save results in AVS file or not. . . . . . . . (avs) = F
     Save results in Visual3 file or not. . . . (visual3) = F
     Save results in binary file or not . . . . . . (bin) = F
     Selected fields :
       Displacement . . . . . . . . . . . . . . . . . . . . = F
       Velocity . . . . . . . . . . . . . . . . . . . . . . = T
       Acceleration . . . . . . . . . . . . . . . . . . . . = F
       Strain . . . . . . . . . . . . . . . . . . . . . . . = F
       Stress . . . . . . . . . . . . . . . . . . . . . . . = F
     Selected components for PostScript snapshots :
       X  . . . . . . . . . . . . . . . . . . . . . . . . . = F
       Y  . . . . . . . . . . . . . . . . . . . . . . . . . = T
       Z  . . . . . . . . . . . . . . . . . . . . . . . . . = F
       Amplitude  . . . . . . . . . . . . . . . . . . . . . = F
```

Printed by Jean Paul Ampuero

Jun 01, 07 1:06 **info** Page 3/4

```
***********************************************
*   I n i t i a l i z a t i o n   p h a s e  *
***********************************************


    Defining the FEM mesh ...... [OK]
    Saving node coordinates in file MeshNodesCoord_sem2d.tab ...... [OK]
    Saving element connectivity in file ElmtNodes_sem2d.tab ...... [OK]

S p e c t r a l   e l e m e n t s   g r i d
===========================================

    Numbering GLL points ...... [OK]
    Total number of GLL points. . . . . . . = 90601

    Saving element/node table in binary file ibool_sem2d.dat ...... [OK]
    Defining nodes coordinates ...... [OK]

    Saving the grid coordinates (coord) in a text file ...... [OK]
    Saving the grid coordinates (coord) in a binary file ...... [OK]

M a t e r i a l   p r o p e r t i e s
=====================================

    Translating input velocity model ...... [OK]
    Defining elasticity work arrays ...... [OK]
    Exporting model ...... [OK]

M e s h   p r o p e r t i e s
=============================

    Checking mesh ...... [OK]
   Max mesh size =  142.616E-03
   Min mesh size =   58.736E-03
   Ratio max/min =    2.428E+00

   RESOLUTION: nodes per min wavelength =    8.000E+00

    Dump PostScript Resolution_sem2d.ps ...... [OK]
    Dump PostScript Stability_sem2d.ps ...... [OK]

T i m e   s o l v e r
=====================

   Time step (secs)      =   17.621E-03
   Number of time steps  = 1987
   Total duration (secs) =   35.013E+00
   Courant number        =  300.000E-03

   STABILITY:  CFL number              =  300.000E-03
    Initializing kinematic fields ...... [OK]
      Max displ =    0.000E+00
      Max veloc =    0.000E+00
    Building the mass matrix ...... [OK]
    Defining boundary conditions ...... [OK]
    Initializing receivers ...

R e c e i v e r s
=================

Receivers have been relocated to the nearest GLL node

Receiver  x-requested  z-requested   x-obtained   z-obtained   distance

       1    0.000E+00    0.000E+00    0.000E+00    0.000E+00    0.000E+00
       2    5.000E+00    0.000E+00    5.000E+00    0.000E+00    0.000E+00
       3   10.000E+00    0.000E+00   10.000E+00    0.000E+00    0.000E+00
       4   15.000E+00    0.000E+00   15.000E+00    0.000E+00    0.000E+00
       5   20.000E+00    0.000E+00   20.000E+00    0.000E+00    0.000E+00
```

Jun 01, 07 1:06 **info** Page 4/4

```
       6   25.000E+00    0.000E+00   25.000E+00    0.000E+00    0.000E+00
       7   30.000E+00    0.000E+00   30.000E+00    0.000E+00    0.000E+00

  Maximum distance between asked and real =   0.000E+00

  Sampling rate (Hz)        =   56.751E+00
  Sampling timestep (secs)  =   17.621E-03
  Total number of samples   = 1988
  Number of receivers       = 7

... [OK]
     Initializing sources ...

 S o u r c e s
 =============

 Sources have been relocated to the nearest GLL node

   Source  x-requested  z-requested   x-obtained   z-obtained     distance

        1    0.000E+00    0.000E+00    0.000E+00    0.000E+00    0.000E+00

  Maximum distance between requested and real =   0.000E+00
... [OK]
Timestep #       0  t =    0.0E+00  vmax =    0.000E+00  dmax =    0.000E+00

Snapshot at timestep = 0


***********************************************
*          S o l v e r   p h a s e            *
***********************************************


---  CPU TIME ESTIMATES (in seconds) :
  CPU time for initialization . . 863.869E-03
  CPU time per timestep . . . . .  14.997E-03
  Total solver CPU time . . . . .  29.799E+00
                   (mins) . . . . 496.651E-03
                   (hours). . . .   8.278E-03

Timestep #    1000  t =   17.6E+00  vmax =   91.661E-03  dmax =   28.653E-03


---  CPU TIME INFORMATION (in seconds) :
  CPU time for initialization . . 863.869E-03
  CPU time per timestep . . . . .  15.302E-03
  Total solver CPU time . . . . .  30.404E+00
                   (mins) . . . . 506.740E-03
                   (hours). . . .   8.446E-03

 Storing sismos data (SEP format) ...


 -------------------------------------------------------------------------------
 -------------------------------------------------------------------------------
 Program  S P E C F E M :  end
 -------------------------------------------------------------------------------
 -------------------------------------------------------------------------------
 Test SH
 -------------------------------------------------------------------------------
 -------------------------------------------------------------------------------
 D a t e : 01 - 06 - 2007                                 T i m e  : 01:06:31
 -------------------------------------------------------------------------------
 -------------------------------------------------------------------------------
```

2.5 Outputs and their visualization

In addition to the screen output described above, `sem2dsolve` generates different files and scripts that allow the user to control the parameters of the simulation and to display the results. All the outputs files follow the naming convention `SomeName_sem2d.xxx`, where `xxx` is one of the following extensions: `tab` for ASCII data files, `txt` for other text files, `dat` for binary data files, etc. This makes it easy to clean a working directory with a single command like `rm -f *_sem2d*`.

2.5.1 Spectral element grid

As explained in the previous section, `sem2dsolve` generates two PostScript files for mesh quality checking purposes: `Stability_sem2d.ps` and `Resolution_sem2d.ps`. The relevant information is contained in the files `Stability_sem2d.tab` and `Resolution_sem2d.tab` and can also be inspected with the Matlab script `PRE/ViewMeshQuality.m`.

2.5.2 Source time function

`sem2dsolve` generates a file called `SourcesTime_sem2d.tab` containing the source time function sampled at the same rate as the receivers. It is important to verify that the spectrum of the source has little power at those high frequencies that are not well resolved by the mesh (those that correspond to less than 5 nodes per wavelength). If this is not the case you must be very cautious in the interpretation of the seismograms in the high frequency range, or low-pass filter the results.

2.5.3 Snapshots

`sem2dsolve` generates snapshots at a constant interval defined by the input parameter `ItSnapshots`. An example is shown in Figure 2.4. Requested fields are exported in binary data files called `xx_XXX_sem2d.dat`, where `xx` is the field code defined in the documentation of the `PLOTS` input block and `XXX` is the 3-digit snapshot number. The user is encouraged to inspect the Matlab script `POST/sample_snapshots.m` to find more about the data formats and their manipulation.

If requested PostScript files `xx_XXX_sem2d.ps` are also dumped. A movie file `movie.gif` can be generated by the script `POST/movie.csh` and displayed by `xanim movie.gif`.

2.5.4 Seismograms

The seismograms are stored using the SEP format, a simple binary block of single precision floats. The components of the vector field (velocity by default) are stored in separate files `U*_sem2d.dat`, where `*` is `x` or `z` in P-SV and `y` in SH. The seismograms header is in the file `SeisHeader_sem2d.hdr`. Its second line contains the sampling timestep `DT`, the number of samples `NSAMP` and the number of stations `NSTA`. The stations coordinates, `XSTA` and `ZSTA`, are listed from the third line to the end of file. With this notations, `U*_sem2d.dat` contains a `NSAMP`×`NSTA` single precision matrix.

You can view the seismograms using any tool that is able to read the SEP format, which is the case of almost all the softwares able to deal with seismic data. `sem2dsolve` generates scripts for the

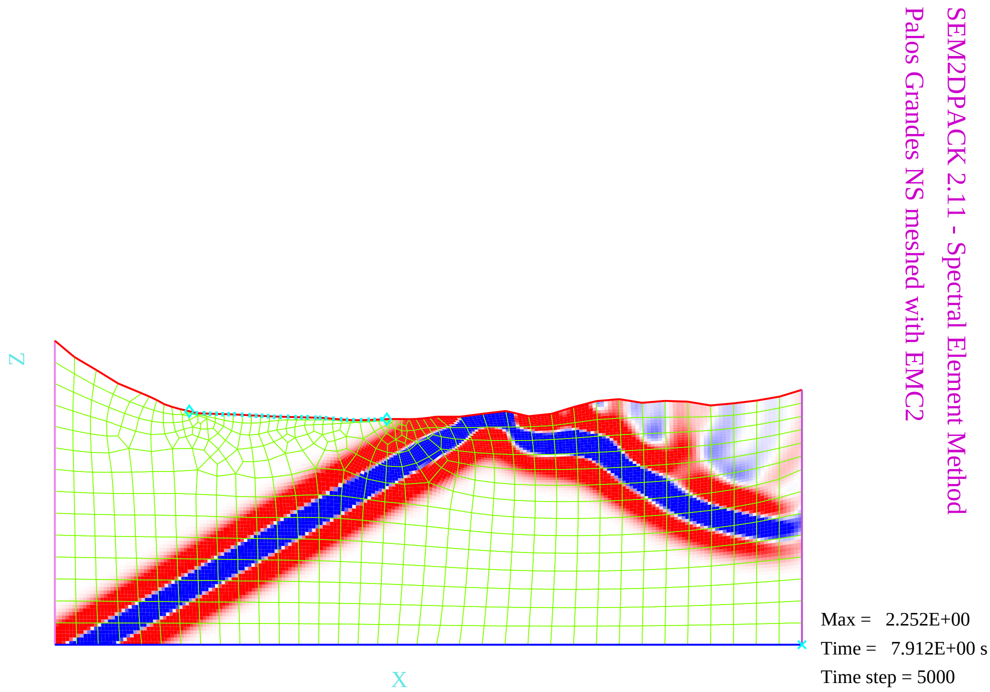

Figure 2.4: Sample snapshot from `EXAMPLES/UsingEMC2/` : an obliquely incident SH plane wave impinging on a sedimentary basin. The unstructured mesh of spectral elements is plotted on background.

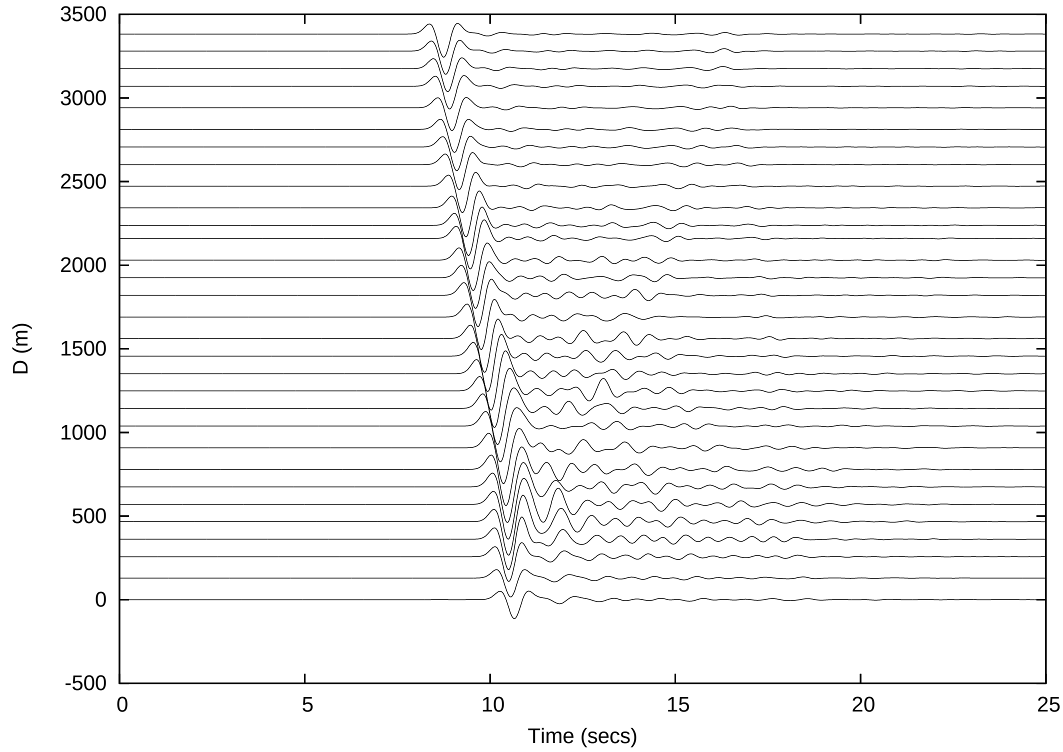

Figure 2.5: Sample seismograms from `EXAMPLES/UsingEMC2/` generated with `POST/seis_plot.csh`.

XSU-Seismic Unix visualization tool[4]:

- `Xline_sem2d.csh` displays all seismograms together on screen
- `PSline_sem2d.csh` plots all seismograms on PostScript files `U*Poly_sem2d.ps`
- `Xtrace_sem2d.csh` prompts the user for a trace number (between 1 and `NSTA`) and then displays this particular trace on screen
- `PStrace_sem2d.csh` does the same as `Xtrace`, but exports the traces as PostScript files `U*TraceXXX_sem2d.ps` where `XXX` is the number of that particular trace

The program `post_seis.exe` performs similar basic manipulation and plotting (through `gnuplot`) of the seismograms. Its interactive menu is self-explanatory. It is usually called inside a script, as in `POST/seis_b2a.csh` (converts all seismograms to ASCII) or `POST/seis_plot.csh` (plots all seismograms together, an example is shown in Figure 2.5).

The script `POST/sample_seis.m` shows how to manipulate and plot seismogram data in Matlab. It uses the functions `POST/read_seis.m` and `POST/plot_seis.m`.

2.5.5 Fault outputs

Fault data from dynamic rupture simulations is stored in `FltXX_sem2d.dat`, where `XX` is the boundary tag of the first side of the fault, `tags(1)` of the `BC_SWFFLT` input block. Every `DELT` seconds (`NSAMP`

[4]Seismic Unix is freely available from the Colorado School of Mines at `http://timna.mines.edu/cwpcodes`

total output times) `NDAT` lines with `NPTS` columns, one per fault node, are written [5]. Stress fields are relative to their initial values, which are contained in the first `NDAT` lines. The header file `FltXX_sem2d.hdr` contains the information needed to read the data file. Its format, line by line, is:

1. `NPTS NDAT NSAMP DELT` (name of parameters)
2. Value of parameters above
3. Name of data fields, separated by ":"
4. `XPTS ZPTS` (name of coordinate axis)
5. from here to the end of file, a two-column table of coordinates of the output fault nodes

The script `FltXX_sem2d.csh` shows how to extract ASCII time series of different fields at given locations on the fault, using Seismic Unix tools.

The program `post_fault.exe` performs basic manipulations of the fault data, including conversion to an ASCII file readable by `gnuplot`. Its interactive menu is self-explanatory.

The script `POST/sample_fault.m` shows how to manipulate and plot fault data in Matlab.

[5] The actual number of columns is `NPTS` +2: Fortran adds a one-word tag at the front and end of each record.

Chapter 3

Use of the mesh generator EMC2

3.1 The mesh generator EMC2

EMC2 is one of the few public domain 2D mesh generators including quadrilateral elements and a Graphical User Interface. It is a C code which sources and executables can be freely downloaded from `http://www-rocq.inria.fr/gamma/cdrom/www/emc2/eng.htm` . Although a complete documentation can be found in that package, we show here an example featuring the most useful functionalities.

Before starting you must provide files containing 2-column data (X,Z), without headers, of all the points needed to define the geometry of the model (topography, sediment bottom).

Once installed, you can run EMC2 by typing `emc2`.

3.2 Notations

The following notations are assumed in this chapter:

- (XXX) = click XXX on top menu bar
- (xxx) = click xxx on bottom menu bar
- <XXX< = click XXX on left menu bar
- >XXX> = click XXX on right menu bar
- xxx = enter xxx from keyboard or from the calculator in the right panel
- "xxx" = type xxx in bottom prompt
- {xxx} = perform action xxx
- *xxx = do xxx as many times as needed
- n*xxx = do xxx n times

3.3 Basic step-by-step

A typical EMC2 session has three steps:

STEP I: CONSTRUCT, defines the geometry of the model

1. Switch to the construction tool:
   <CONSTRUCTION<
2. Load the points:
   (POINT) (xy file) "palosgrandes.dat"
   You must give the full path to your points-file, the root directory being the one where you launched `emc2`.
3. Reset the figure window to fit all points:
   >SHOW ALL>

   The original data has some geometrical features that are too complex to be meshed by quadrilaterals, for instance the corners at the N and S ends of the basin, you may want to smooth out these features. You also need to define the extreme boundaries of the region to be modelled (N,S and bottom absorbing boundaries) and some additional points on the free surface outside the basin. You must modify the data set (add and delete points):
4. Add new points:
   a. with the mouse:
      (POINT) (mouse) *{click in figure window}
   b. by coordinates:
      (POINT) (xy pt) *{ $x=y=$ }
      This is the safest way to get really vertical and horizontal boundaries needed for the absorbing conditions in SPECFEM90. You probably need to get the coordinates of an existent reference point:
      (POINT) <QUERY< (point) *{click on point}
   c. you can also reload another point-file (I2)
5. Delete points,
   (POINT) <DESTRUCT< (point) *{click on point}

   Now you must define the geometry of the domains. These macro-blocks are intended to be internally meshed by deformed quadrilaterals. Their geometry follows the geometry of the geological model (one domain per material). Each domain must be bounded by segments or splines:
6. Segments:
   (SEGMENT) (point) 2*{click extreme point}
7. Splines:
   (SPLINE) (point) *{click point}
   You will see the spline evolve as you click points.

STEP II: PREPARE, defines the properties of the discrete spectral element mesh

1. Switch to the preparing mesh tool:
   <PREP MESH<

2. Define domains with rock n:
   (DOMAIN REF) $n=$ (any) *{click inside domain}
   You will see the domains edges get colored and the domains get numbered with n.
3. At any moment you can decide to show or not the domain decomposition:
   To hide the domain decomposition:
   >REFRESH>
   Show the domain decomposition:
   (SHOW) (ALL)
4. Remove a domain definition:
   (REMOVE) (DOMAINE) (any) {click inside domain}
   WARNING: corrections to the domain decomposition are sometimes displayed only after refreshing the figure window.
5. Now you must define the subdivision of each domain in quadrilateral finite elements. Define the number n of elements on each edge:
   (NB INTERVAL) $n=$ (any) {click edge}
   You will see the intermediate points appear. The number of intervals n is mainly dictated by the resolution criterion: elements should be smaller than the smallest wavelength you want to propagate. Moreover, **a domain can be quadrangulated only if the total number of intervals along its perimeter is even** (the sum of all n along its boundaries). However, a quality mesh is not always guaranteed and you need to proceed by trial and error (emc2 allows you to jump back and forth between the different steps of the meshing procedure).
6. Finally you must define the external boundaries of the modelled region which will have a special treatment. You must associate a tag (a number) to each absorbing boundary. No convention is assumed but you should remember those tags later when setting the boundary conditions in SEM2D. It is also useful to assign a tag to the free surface boundary, that will be eventually used by SEM2D to locate the receivers or sources.
   Define a boundary with index n:
   (LINE REF) $n=$ (any) *{click edge}
   Of course each boundary can be composed of many domain edges. Refresh the display to better see the boundaries. *The same procedure applies to define split-node interfaces such as faults and cracks: you must assign a different tag to each side of the fault.*
7. Save your work in EMC2 format:
   <SAVE< "name"
   The resulting file is `name.emc2_bd`

STEP III: EDIT, generates the mesh

1. Switch to the edit mesh tool:
   <EDIT_MESH<
   Press ENTER 4 times.
   A triangles mesh appears. You must convert it to a quad mesh:
2. Convert the triangle mesh to a quad mesh:
   <QUADRANGULATE> <ALL>
   You can smooth the mesh with: <REGULARIZE> *<ALL>

   The final mesh is displayed. If there remain some triangles come back to the previous step and figure out how to modify the points per edge to help the mesher. Some experience is needed here.

3. Renumber the mesh, in order to optimize computations:
   *<RENUMBER>
4. Define the boundary condition for the 4 corner nodes of the model: (these nodes belong to 2 external boundaries so they were given a reference number =0)
   (MODIF_REF) $n=$ (corner) {click close to corner, inside element}
   Where n is the reference number of one of the 2 boundaries containing the corner node. Zooming can be useful. The same operation must be performed for the corner nodes of the subdomains belonging to an external boundary, and for the the crack tip nodes. *However, as a special case, crack tip nodes must be assigned the -1 tag.*
5. Export the mesh:
   <SAVE<
   Two questions are asked in the bottom prompt:
   - Format of the file, you must select:
     "ftq"
   - Prefix name for the file
     "name"
     The resulting file name will be `name.ftq`

3.4 Some additional tips

- Whenever possible it is better to mesh a domain with a *structured* mesh (a deformed cartesian grid). This can be done with (QUADRANGULATE), during the PREPARE step. See our FAQ for further details.
- To load an existent project, in the construction tool or in the preparation mesh tool:
  <RESTORE< "name"
  EMC2 will look for the file `name.emc2_bd`. Beware: the project loaded will replace the actual project if any, there is no superposition.
- BUG WARNING (13/07/01): the Sun release of EMC2 has a bug with the reference indices in the ftq format This bug is fixed in the 2.12c version. If you work on a Sun station, download the most recent version of the sources, rather than the executable, and compile it yourself.
- To densify (h-refinement) an existent mesh use the script `SEM2DPACK/POST/href.csh`. It edits the `*.emc2_bd` file, then you can restore it in EMC2 and save it in `*.ftq` format.
- To create a fault, in EDIT_MESH mode:
  a. Crack an existent edge:
     (CRACK) (segment)
  b. Give a reference number to each side of the fault :
     (MODIF_REF) $n=$ (segment)
  c. Give the tag "-1" to crack tip nodes:
     (MODIF_REF) $-1=$ (corner) *{click close to crack tip node, inside element}
- Note that only Q4 elements (4 control nodes) are supported. For a smoother description of boundaries Q9 would be desirable.

Chapter 4

Frequently Asked Questions

4.1 SEM2D

Segmentation fault

This problem is often related to a small stack size in your computer settings. In your Linux shell do: `ulimit -s unlimited`. Place this command in your `.bashrc`.

4.2 EMC2

I can't get rid of a few triangles

Obtaining a quality quad mesh is not always a trivial task. Trial and error and experience is needed. This can be by far the most time consuming stage of modeling.

First make sure that the total number of element edges along the perimeter of each mesh domain is even. This is a necessary topological condition to generate a quad-only mesh.

When the geometry seems too complicated for quad meshing you should consider simplifying the geometry, especially those details that are much smaller than the dominant wavelength.

If the above fails or does not apply, you have to help the mesher. The recommended procedure in EMC2 is:

1. Divide your original mesh into simple domains, in such a way that *most* domains have exactly four sides (possibly curved) and the remaining non-four-sided domains are as small as possible.
2. Generate a structured quad-mesh (a regular grid) inside each four-sided domain with the (QUADRANGULATE) tool of the PREP_MESH mode, as described in section 5.2.13 of EMC2's manual (note that this is *not* the same as the <QUADRANGULATE> button in the EDIT_MESH mode).
3. Proceed as usual (triangulation followed by quadrangulation) inside the remaining non-four-sided domains. If these are small enough EMC2 should not have problems doing a correct tri-to-quad meshing.

Bibliography

Ampuero, J.-P. (2002). *Etude physique et numérique de la nucléation des séismes*. Ph. D. thesis, Université Paris 7, Denis Diderot, Paris.

Andrews, D. (1999). Test of two methods for faulting in finite difference calculations. *Bull. Seis. Soc. Am. 89*, 931–937.

Komatitsch, D. (1997). *Méthodes spectrales et éléments spectraux pour l'équation de l'élastodynamique 2D et 3D en milieu hétérogène*. Ph. D. thesis, Institut de Physique du Globe de Paris, Paris.

Komatitsch, D. and J. Tromp (1999). Introduction to the spectral-element method for 3-D seismic wave propagation. *Geophys. J. Int. 139*, 806–822.

Komatitsch, D. and J. P. Vilotte (1998). The spectral element method: an efficient tool to simulate the seismic response of 2D and 3D geological structures. *Bull. Seis. Soc. Am. 88*, 368–392.

Komatitsch, D., J. P. Vilotte, R. Vai, and F. J. Sánchez-Sesma (1999). The Spectral Element method for elastic wave equations: application to 2D and 3D seismic problems. *Int. J. Num. Meth. Engng 45*(9), 1139–1164.

Patera, A. (1984). A spectral element method for fluid dynamics: laminar flow in a channel expansion. *J. Comp. Phys. 54*, 468–488.

Vai, R., J. M. Castillo-Covarrubias, F. J. Sánchez-Sesma, D. Komatitsch, and J. P. Vilotte (1998). Elastic wave propagation in an irregularly layered medium. *Soil Dyn. Earthquake Eng 18*(1), 11–18.