
SEM2DPACK

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

User's Guide

Version 2.3.6

August 2009

Jean-Paul Ampuero

California Institute of Technology
Seismological Laboratory
1200 E. California Blvd., MC 252-21
Pasadena, CA 91125-2100, USA
E-mail: ampuero@gps.caltech.edu
Web: <http://web.gps.caltech.edu/~ampuero>

Phone: (626) 395-6958

Fax : (626) 564-0715

Contents

1	Introduction	5
1.1	Overview	5
1.2	History and credits	6
1.3	Download and updates	6
1.4	Requirements	7
1.5	Installation	7
1.6	Documentation	7
1.7	Support	8
1.8	Contributions	8
1.9	License	9
2	Physical background	10
2.1	General assumptions and conventions	10
2.2	Material rheologies	11
2.2.1	Linear elasticity	11
2.2.2	Linear visco-elasticity	12
2.2.3	Coulomb plasticity and visco-plasticity	12
2.2.4	Continuum damage	13
2.3	Boundary conditions	14
2.3.1	Absorbing boundaries	14
2.4	Fault interface conditions	15
2.4.1	Linear slip law	15
2.4.2	Normal stress response	15
2.4.3	Friction	16
3	Mesh generation	18
3.1	General guidelines	18
3.2	Meshing features included in the solver	19
3.3	Meshing with the MESH2D Matlab utilities	19
3.4	Generating a mesh with EMC2	20
3.4.1	The mesh generator EMC2	20
3.4.2	Notations	20
3.4.3	Basic step-by-step	21
3.4.4	Further tips	24
4	The solver SEM2D	25

4.1	About the method	25
4.2	Basic usage flow	26
4.3	General format of the input file	26
4.4	Input Blocks Reference Guide	30
4.5	Verifying the settings and running a simulation	51
4.6	Outputs, their visualization and manipulation	56
4.6.1	Spectral element grid	56
4.6.2	Source time function	56
4.6.3	Snapshots	56
4.6.4	Seismograms	58
4.6.5	Fault outputs	58
4.6.6	Stress glut	60
4.6.7	Energies	60
4.6.8	Matlab utilities	60
5	Adding features to SEM2D (notes for advanced users)	62
5.1	Overview of the code architecture	62
5.2	Accessible areas of the code	62
6	Frequently Asked Questions	64
6.1	SEM2D	64
6.2	EMC2	64

Chapter 1

Introduction

1.1 Overview

The SEM2DPACK package is a set of software tools for the simulation and analysis of 2D wave propagation and dynamic fracture, with emphasis on computational seismology and earthquake dynamics. The core of the package is SEM2D, a solver for the 2D elastic wave equations and dynamic earthquake rupture based on the Spectral Element Method (SEM) with explicit time stepping. Chapter 2 of this User's Guide summarizes the range of problems that can be solved with SEM2DPACK. Section 4.1 provides some background on the SEM. The essential properties of the method are its high order accuracy, affordable at competitive computational cost, its geometrical flexibility to treat realistic, complicated crustal structures, and its natural treatment of mixed boundary conditions such as fault friction.

SEM2DPACK provides tools for each step of the general flow of a simulation project:

1. **Mesh generation:** partition the domain into (deformed) quadrilateral elements. Whereas no general mesh generation code is included, SEM2DPACK contains basic meshing utilities for structured and semi-structured grids and can import unstructured quadrilateral meshes generated externally. These features are described in Chapter 3.
2. **Mesh quality verification:** check the accuracy, stability and computational cost, applying tools described in Section 4.5. Return to previous step if needed.
3. **Numerical simulation:** run the SEM2D solver. Chapter 4 explains its usage.
4. **Post-processing:** visualization and analysis of the output. A number of post-processing and graphic tools are included, as described in Section 4.6. Outputs are in the form of binary data files, ASCII data files and PostScript figure files. Scripts are provided for graphic display and analysis on Seismic Unix, Gnuplot and Matlab. We recommend the usage of the Matlab functions included (see Section 4.6.8).

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, bug reports (see Section 1.7) and contributions to the code itself (see Section 1.8).

1.2 History and credits

The main part of the elastic-isotropic solver was written by Dimitri Komatitsch (Komatitsch, 1997) as part of his Ph.D. thesis under the direction of Prof. Jean-Pierre Vilotte at the Institut de Physique du Globe de Paris (IPGP). The elastic-anisotropic solver and several significant improvements were added by D. Komatitsch later as part of a research contract with DIA Consultants. Further functionalities were added by Jean-Paul Ampuero, also as part of a Ph.D. thesis (Ampuero, 2002) directed by Prof. Vilotte at IPGP. 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 version 2.0, most of the solver was rewritten in preparation for the implementation of higher level functionalities. Developments for the simulation of earthquake dynamics (Ampuero, 2002) were included in the main branch of SEM2DPACK in October 2003 (version 2.2). Spontaneous rupture along multiple non-planar faults can be currently modelled, with a range of friction laws.

Non-linear, inelastic materials were introduced in March 2008 (version 2.3). Damage and visco-plastic rheologies are included especially for the modeling of earthquake rupture with off-fault dissipation.

SEM2DPACK has been applied in work related to the following publications:

- Madariaga *et al.* (2006): dynamic rupture and seismic wave radiation on faults with geometrical complexities (kinks)
- Haney *et al.* (2007): fault reflections from fluid-infiltrated faults
- De la Puente *et al.* (2007): as benchmark method for anisotropic wave propagation
- Kaneko *et al.* (2008): dynamic earthquake rupture with rate-and-state friction
- Lyakhovsky *et al.* (2009): non-linear wave propagation in damaged rocks

1.3 Download and updates

SEM2DPACK is hosted by SourceForge at

<http://sourceforge.net/projects/sem2d/>.

All versions of the code can be downloaded from the package repository at

http://sourceforge.net/project/showfiles.php?group_id=182742.

Taking full advantage of the convenient features offered by SourceForge (subscribe to new release announcements, submit and track bug reports) requires a SourceForge.net account,

which can be created at

<http://sourceforge.net/account/registration/>.

SEM2DPACK is updated regularly, typically every one or two months. To receive email notifications about new releases you must sign up for the “Package Monitor” at

http://sourceforge.net/project/filemodule_monitor.php?filemodule_id=212397.

1.4 Requirements

Compiling the solver code requires the `make` utility and a Fortran 95 compiler. The code is being developed with the Intel compiler for Linux. It works properly with the Intel compiler starting with version 8.0.046_pe047.1, so make sure you have a recent version of `ifort`. Other compilers are not being tested on a regular basis, so please report any related problems.

The solver runs under the Linux operating system. In particular, input/output file name conventions are specific to Linux. Other operating systems have not been tested.

Pre-processing and post-processing tools, including graphic visualization, are provided for Seismic Unix, Gnuplot, GMT and Matlab. The included Matlab tools are by far the most complete, so a Matlab license is highly recommended. Matlab “clone” softwares have not been tested.

1.5 Installation

1. Uncompress and expand the SEM2DPACK package: `tar xvfz sem2dpack.tgz`
2. Go to the source directory: `cd SEM2DPACK/SRC`
3. Edit the `Makefile` according to your FORTRAN 95 compiler, following the instructions therein.
4. Modify the optimization parameters declared and described in `SRC/constant.f90`.
5. Compile: `make`
6. 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/`.

1.6 Documentation

Documentation is available through the following resources:

- This User’s Manual
- The `EXAMPLES` directory contains several examples, some have a `README` file
- The pre-processing and post-processing tools for Matlab are documented through Matlab’s help. For instance `help mesh2d` provides an overview of the MESH2D utilities, and `help mesh2d_wedge` provides detailed documentation for the wedge meshing function
- The `ToDo` file contains a list of known issues

1.7 Support

Support for users of SEM2DPACK is available through a *tracking system* at

http://sourceforge.net/tracker/?group_id=182742,

Three separate *tracker lists* deal with the following aspects:

- *Feature Requests*: requests for implementation of new features
- *Support Requests*: questions related to the usage of SEM2DPACK
- *Bugs*: bug reports

Before submitting an issue make sure that:

1. you read the documentation (see Section 1.6), including the Frequently Asked Questions (Chapter 6). Suggestions on how to improve the documentation are also welcome.
2. you are running the most recent version of SEM2DPACK. Your issue might have been already fixed in a more recent version.
3. you understand the changes listed in SEM2DPACK’s `ChangeLog` file, especially changes in the format of the input files
4. your problem has not been treated in previous submissions. The tracker lists are browsable and searchable. By default the browser shows only pending issues, to see the complete list (including resolved issues) set “Status” to “Any”.

A new submission must include the input files needed to reproduce your problem (`Par.inp`, `*.ftq`, `*.mesh2d`, 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.

1.8 Contributions

Contributions to SEM2DPACK by experienced programmers are always welcome and encouraged. Although the code is stable for typical applications in computational seismology

and earthquake dynamics, 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 solver code is written in FORTRAN 95. Resources available for programmers include:

- A `ToDo` file included with SEM2DPACK contains a wish list that ranges from basic functionalities to complex code re-engineering.
- Chapter 5 gives some guidelines for programmers.
- A “Developers Forum” to discuss the implementation of new features is available at http://sourceforge.net/forum/forum.php?forum_id=635737,

1.9 License

This software is freely available for academic research purposes. If you use this software in writing scientific papers include proper attributions to its author, Jean-Paul Ampuero.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

Chapter 2

Physical background

This chapter summarizes the physical assumptions and notations in SEM2DPACK. Footnotes provide reference to the input arguments described in Chapter 4.

2.1 General assumptions and conventions

The coordinate system is Cartesian (rectangular). SEM2DPACK works in the two-dimensional (x, z) plane, where x is the horizontal coordinate, with positive direction pointing to the right, and z is the vertical coordinate, with positive direction pointing upwards. The coordinates (x, y, z) will be also denoted as (x_1, x_2, x_3) . This notation carries also for subscripts. For instance, the k -th component of displacement is denoted as u_k , with $k = 1, 2, 3$ or with $k = x, y, z$.

The reference frame is Eulerian. Infinitesimal strain is assumed. The (symmetric) infinitesimal strain tensor ϵ is defined as

$$\epsilon_{ij} = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \quad (2.1)$$

Material density is denoted $\rho(x, z)$. The displacements and stresses relative to an initial equilibrium configuration are denoted $u_k(x, z, t)$ and $\sigma_{ij}(x, z, t)$, respectively. External forces (sources) are denoted $f_i(x, z, t)$. SEM2DPACK solves the following equations of motion to obtain the relative displacements $u_k(x, z, t)$:

$$\rho \frac{\partial^2 u}{\partial t^2} = \frac{\partial \sigma_{ij}}{\partial x_j} + f_i \quad (2.2)$$

where summation over repeated indices is assumed. The initial conditions are $u_k = 0$ and $\partial u_k / \partial t = 0$. Stresses are related to strain, and possibly to other internal variables, by constitutive equations described in Section 2.2. The governing equations are supplemented by boundary conditions, described in Section 2.3. SEM2DPACK actually solves the governing

equations in variational (weak) form, as described in any textbook on the finite element method.

Two types of 2D problems are solved¹:

- *Plane strain*: Also known in seismology as P-SV, and in fracture mechanics as inplane mode or mode II. It is assumed that $u_3 = 0$ and $\partial/\partial x_3 = 0$. Hence, $\epsilon_{13} = \epsilon_{23} = \epsilon_{33} = 0$ and there are two degrees of freedom per node, u_x and u_z .
- *Antiplane shear*: Also known in seismology as SH, and in fracture mechanics as antiplane mode or mode III. It is assumed that $u_1 = u_2 = 0$ and $\partial/\partial x_3 = 0$. Hence, only ϵ_{13} and ϵ_{23} are non-zero and there is one degree of freedom per node, u_y .

2.2 Material rheologies

We describe here the constitutive equations implemented in SEM2DPACK, relating stress (σ_{ij}), strain (ϵ_{ij}) and internal variables.

2.2.1 Linear elasticity

Linear isotropic elasticity

Stress and strain are linearly related by Hooke's law, $\sigma_{ij} = c_{ijkl}\epsilon_{ij}$, where c_{ijkl} is the tensor of elastic moduli. In particular, for isotropic elasticity:

$$\sigma_{ij} = \lambda \epsilon_{kk} \delta_{ij} + 2\mu \epsilon_{ij} \quad (2.3)$$

where λ and μ are Lamé's first and second parameters, respectively. In 2D plane strain the only relevant stress components are σ_{11} , σ_{22} and σ_{12} . The intermediate stress σ_{33} , although not null, does not enter in the equations of motion. The S and P wave speeds are $c_S = \sqrt{\mu/\rho}$ and $c_P = \sqrt{(\lambda + 2\mu)/\rho}$, respectively. In 2D antiplane shear only the stress components σ_{13} and σ_{23} are relevant, and only S waves are generated.

Linear anisotropic elasticity

Transverse anisotropy with vertical symmetry axis is implemented for 2D plane strain (Komatitsch *et al.*, 2000). The stress-strain constitutive relation is:

$$\begin{pmatrix} \sigma_{xx} \\ \sigma_{zz} \\ \sigma_{xz} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{13} & 0 \\ c_{13} & c_{33} & 0 \\ 0 & 0 & c_{55} \end{pmatrix} \begin{pmatrix} \epsilon_{xx} \\ \epsilon_{zz} \\ 2\epsilon_{xz} \end{pmatrix} \quad (2.4)$$

where the c_{ij} are elastic moduli.

¹In the `&GENERAL` input block, plane strain is selected by `ndof=2` and antiplane shear by `ndof=1`.

2.2.2 Linear visco-elasticity

Generalized Maxwell material

Not implemented yet.

Kelvin-Voigt material

Kelvin-Voigt damping can be combined with any of the other constitutive equations by replacing the elastic strain ϵ by $\epsilon^* = \epsilon + \eta \partial \epsilon / \partial t$, where η is a viscosity timescale.

The resulting quality factor Q is frequency-dependent, $Q^{-1}(f) = 2\pi\eta f$. This rheology is not appropriate to model crustal attenuation with constant Q , unless the source has a narrow frequency band and η is selected to achieve a given Q value at the dominant frequency of the source.

The main application of Kelvin-Voigt viscosity is the artificial damping of high-frequency numerical artifacts generated by dynamic faults. Dynamic source simulations using methods that discretize the bulk, such as finite difference, finite element and spectral element methods, are prone to high frequency numerical noise when the size of the process zone is not well resolved. Efficient damping is typically achieved by a thin layer of Kelvin-Voigt elements surrounding the fault with $\eta/\Delta t = 0.1$ to 0.3 and a layer thickness of 1 to 2 elements on each side of the fault.

2.2.3 Coulomb plasticity and visco-plasticity

Perfect plasticity

Perfect plasticity with a Coulomb yield function is implemented for 2D plane strain, as in Andrews (2005).

The total strain is the sum of an elastic and a plastic contribution, $\epsilon = \epsilon^e + \epsilon^p$. The plastic strain is assumed to be purely deviatoric ($\epsilon_{kk}^p = 0$). Plastic yield occurs when the maximum shear stress over all orientations,

$$\tau_{max} = \sqrt{\sigma_{xz}^2 + (\sigma_{xx} - \sigma_{zz})^2/4}, \quad (2.5)$$

reaches the yield strength,

$$Y = c \cos(\phi) - (\sigma_{xx} + \sigma_{zz}) \sin(\phi)/2, \quad (2.6)$$

where c is the cohesion and ϕ is the internal friction angle.

Visco-plasticity

In classical Duvaut-Lions visco-plasticity the (visco-)plastic strain rate is proportional to the excess of stress over the yield strength:

$$\dot{\epsilon}_{kl}^p = \frac{1}{2\mu T_v} \langle \tau_{max} - Y \rangle \frac{\tau_{ij}}{\tau_{max}} \quad (2.7)$$

where T_v is the visco-plastic relaxation time, $\langle x \rangle \doteq (x + |x|)/2$ is the ramp function and $\tau_{ij} = \sigma_{ij} - \frac{1}{3}\sigma_{kk} \delta_{ij}$ is the deviatoric part of the stress tensor.

Visco-plasticity is often employed as a regularization of plasticity to avoid or delay the occurrence of strain localization features, such as shear bands, that are mesh-dependent. For that particular application, T_v is typically set to the average P wave traveltime across a few grid points, i.e. a few times the average spacing between GLL nodes divided by the P wave speed.

2.2.4 Continuum damage

The continuum damage formulation by Lyakhovsky *et al.* (1997), including damage-related plasticity as introduced by Hamiel *et al.* (2004), is implemented with modifications for 2D plane strain.

The first and second invariants of the 2D elastic strain tensor are defined as $I_1 = \epsilon_{kk}^e$ and $I_2 = \epsilon_{ij}^e \epsilon_{ij}^e$, respectively. A strain invariant ratio is defined as $\xi = I_1/\sqrt{I_2}$. The following non-linear stress-strain relation is assumed (Lyakhovsky *et al.*, 1997, eq. 12):

$$\sigma_{ij} = (\lambda - \gamma/\xi) I_1 \delta_{ij} + (2\mu - \gamma\xi) \epsilon_{ij}^e \quad (2.8)$$

where γ is an additional elastic modulus. The elastic moduli depend on a scalar damage variable, $0 \leq \alpha \leq 1$, through (Lyakhovsky *et al.*, 1997, eq. 19):

$$\lambda = \lambda_0 \quad (2.9)$$

$$\mu = \mu_0 + \gamma_r \xi_0 \alpha \quad (2.10)$$

$$\gamma = \gamma_r \alpha \quad (2.11)$$

where λ_0 and μ_0 are Lamé's parameters for the intact material ($\alpha = 0$). The parameter ξ_0 is the threshold value of the strain invariant ratio ξ at the onset of damage. It is related to the internal friction angle ϕ in a cohesionless Mohr-Coulomb yield criterion by the 2D plane strain version of Lyakhovsky *et al.* (1997, eq. 37):

$$\xi_0 = \frac{-\sqrt{2}}{\sqrt{1 + (\lambda_0/\mu_0 + 1)^2 \sin^2 \phi}} \quad (2.12)$$

The scaling factor γ_r is chosen such that convexity is lost at $\alpha = 1$ when $\xi = \xi_0$. It is derived from the 2D plane strain version of Lyakhovsky *et al.* (1997, eq. 15):

$$\gamma_r = p + \sqrt{p^2 + 2\mu_0 q} \quad (2.13)$$

where

$$q = (2\mu_0 + 2\lambda_0)/(2 - \xi_0^2) \quad (2.14)$$

$$p = \xi_0(q + \lambda_0)/2 \quad (2.15)$$

The evolution equation for the damage variable is (Lyakhovsky *et al.*, 1997, eq. 20)

$$\dot{\alpha} = \begin{cases} C_d I_2(\xi - \xi_0) & \text{if } \xi > \xi_0 \\ 0 & \text{otherwise} \end{cases} \quad (2.16)$$

No healing is assumed below ξ_0 . The evolution of the plastic strain ϵ_{ij}^p is driven by the damage variable α (Hamiel *et al.*, 2004, eq. 9):

$$\dot{\epsilon}_{ij}^p = \begin{cases} \tau_{ij} C_v \dot{\alpha} & \text{if } \dot{\alpha} \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad (2.17)$$

where $\tau_{ij} = \sigma_{ij} - \frac{1}{3}\sigma_{kk}\delta_{ij}$ is the deviatoric part of the stress tensor. The parameter C_v is of order $1/\mu_0$ and² is related to the seismic coupling coefficient $0 < \chi < 1$ by (Ben-Zion and Lyakhovsky, 2006)

$$C_v = \frac{1 - \chi}{\chi} \frac{1}{\mu_0} \quad (2.18)$$

2.3 Boundary conditions

2.3.1 Absorbing boundaries

Two approximate absorbing boundary conditions (ABC) are implemented to model the outwards propagation of waves at the boundaries of the computational domain. Both conditions are of paraxial type. Their performance is appropriate at normal incidence but degrades at grazing incidence.

Clayton-Engquist ABC

In the local coordinate frame (t, n) related to the tangential (t) and outgoing normal (n) directions to the boundary, the first-order accurate ABC proposed by Clayton and Engquist (1977) reads:

$$\dot{u}_t = -c_S \frac{\partial u_t}{\partial x_n} \quad (2.19)$$

$$\dot{u}_n = -c_P \frac{\partial u_n}{\partial x_n} \quad (2.20)$$

The implementation is based on an equivalent formulation as a mixed boundary condition that relates tractions T to displacements u :

$$T_t = -\rho c_S \dot{u}_t \quad (2.21)$$

$$T_n = -\rho c_P \dot{u}_n \quad (2.22)$$

The formulation above is for P-SV mode. In SH mode the ABC is $T_y = -\rho c_S \dot{u}_y$.

²In `&MAT_DM3`, the input argument `R` is defined as $R = \mu_0 C_v$.

Stacey ABC

The second-order accurate ABC introduced by Stacey (1988) under the name “P3” is:

$$\dot{u}_t = -c_S \frac{\partial u_t}{\partial x_n} - (c_P - c_S) \frac{\partial u_n}{\partial x_t} \quad (2.23)$$

$$\dot{u}_n = -c_P \frac{\partial u_n}{\partial x_n} - (c_P - c_S) \frac{\partial u_t}{\partial x_t} \quad (2.24)$$

Its formulation as a mixed boundary condition is:

$$T_t = -\rho c_S \dot{u}_t + \rho c_S (2c_S - c_P) \frac{\partial u_n}{\partial x_t} \quad (2.25)$$

$$T_n = -\rho c_P \dot{u}_n - \rho c_S (2c_S - c_P) \frac{\partial u_t}{\partial x_t} \quad (2.26)$$

This ABC is only implemented in P-SV mode.

2.4 Fault interface conditions

2.4.1 Linear slip law

Represents a compliant fault zone with elastic contact. See Haney *et al.* (2007). [...]

2.4.2 Normal stress response

Unilateral contact

No interpenetration during contact, free stress during opening. [...]

Modified Prakash-Clifton regularization

Regularization of the normal stress response, as required for bimaterial rupture problems, is implemented following Rubin and Ampuero (2007). The frictional strength is proportional to a modified normal stress σ^* , related to the real fault normal stress, σ , by either of the following evolution laws:

- Version with a regularization *slip* scale:

$$\dot{\sigma}^* = \frac{|V| + V^*}{L_\sigma} (\sigma - \sigma^*) \quad (2.27)$$

where V is slip rate, and V^* and L_σ are constitutive parameters³.

³In `&BC_DYNFLT_NOR`, this law is set by `kind=2`, and the two relevant parameters are `V` and `L`.

- Version with a regularization *time* scale:

$$\dot{\sigma}^* = \frac{1}{T_\sigma} (\sigma - \sigma^*) \quad (2.28)$$

where T_σ is a constitutive parameter⁴.

2.4.3 Friction

Slip-weakening friction

Slip occurs when the fault shear stress reaches the shear strength $\tau = \mu\sigma$ (or $\tau = \mu\sigma^*$ if the Prakash-Clifton law is assumed). [...] The friction coefficient μ is a function of the cumulated slip D , given by one of the following laws:

- Linear slip-weakening law:

$$\mu = \max \left[\mu_d, \mu_s - \frac{\mu_s - \mu_d}{D_c} D \right] \quad (2.29)$$

- Exponential slip-weakening law:

$$\mu = \mu_s - (\mu_s - \mu_d) \exp(-D/D_c) \quad (2.30)$$

“Fast” rate-and-state-dependent friction

Friction with fast (power law) velocity weakening at fast slip speed is a first order proxy for physical weakening processes that operate on natural fault zones at coseismic slip velocities. A rate-and-state dependent friction law with fast velocity-weakening is implemented in SEM2DPACK, similar to that adopted e.g. by Ampuero and Ben-Zion (2008). The friction coefficient depends on slip velocity (V) and a state variable (θ):

$$\mu_f = \mu_s + a \frac{V}{V + V_c} - b \frac{\theta}{\theta + D_c}. \quad (2.31)$$

The state variable has units of slip and is governed by the evolution equation

$$\dot{\theta} = V - \theta V_c / D_c. \quad (2.32)$$

The friction law is defined by the following constitutive parameters: μ_s is the static friction coefficient, a and b are positive coefficients of a direct effect and an evolution effect, respectively, V_c is a characteristic velocity scale⁵, and D_c is a characteristic slip scale.

The steady-state ($\dot{\theta} = 0$) friction coefficient

$$\mu_f = \mu_s + (a - b) \frac{V}{V + V_c} \quad (2.33)$$

⁴In `&BC_DYNFLT_NOR`, this law is set by `kind=3`, and the relevant parameter is `T`.

⁵`vstar` in `&BC_DYNFLT_RSF`

weakens asymptotically as $1/V$ when $V \gg V_c$, if $a < b$, approaching its dynamic value ($\mu_d = \mu_s + a - b$) over a relaxation timescale D_c/V_c . The value of the relaxation time D_c/V_c tunes the weakening mechanism between two limit cases: slip-weakening and velocity-weakening. If D_c/V_c is much longer than the typical time scale of fluctuation of the state variable ($\approx \theta/\dot{\theta}$), Equation 2.32 becomes $\dot{\theta} \approx V$, implying that θ is proportional to slip and that the evolution term of the friction coefficient is effectively slip-weakening, with characteristic slip-weakening distance D_c . Conversely, if D_c/V_c is short the relaxation to steady state is fast, $\theta/D_c \approx V/V_c$ and the friction is effectively velocity-weakening, with characteristic velocity scale V_c .

Logarithmic rate-and-state friction

Not implemented yet.

Dieterich and Ruina classical rate-and-state laws, with aging or slip state evolution law.

Chapter 3

Mesh generation

3.1 General guidelines

The Spectral Element Method (SEM) requires an initial decomposition of the space domain into quadrilateral elements (a quad mesh). Obtaining the best performance (accuracy/cost) out of the SEM imposes constraints on the mesh design:

- The interfaces between different materials, at which sharp contrasts of material properties occur, should *preferably* coincide with faces of the elements. This is sometimes called an *adapted mesh* and is the only way to preserve spectral accuracy at material interfaces.
- Fault planes, across which displacement discontinuities occur, *must* coincide with element faces. Faults are implemented with a *split node* formulation.
- Elements can be deformed, but extremely small and extremely large angles between faces of a same element must be avoided. This would penalize both accuracy and stability.
- The linear size of the elements must be small enough, so that each element contains enough computational nodes per minimum wavelength, and each fault boundary element contains enough nodes per rupture process zone.
- Unnecessarily small elements should be avoided, they penalize the stability of the method.

Generating high quality quad meshes for complicated geological models is not yet a fully automated process, and can be very time-consuming. Iterations between mesh generation and mesh quality check are sometimes required. The last two constraints above are addressed more quantitatively in Section 4.5. Mesh quality assessment tools are also presented in Section 4.5.

The remainder of this chapter describes three possible ways to generate quad meshes, by order of complexity:

1. If the geometrical structure of the model is simple or if the user prefers to sacrifice accuracy by using a non-adapted structured mesh, i.e. a logically cartesian mesh where the element faces do not necessarily follow the material interfaces, the basic built-in meshing capabilities of the solver SEM2D, described in Section 3.2, are sufficient.

2. Moderately complicated meshes can be generated with the included Matlab tools, described in Section 3.3.
3. Adapted meshes for more complicated geological models must be generated with some external software. As an illustration, the usage of the freely available 2D mesh generation software EMC2 is described in Section 3.4.

SEM2DPACK provides only basic meshing capabilities and does not include an unstructured mesh generator for complicated, realistic geological models. This chapter describes how to achieve that task with an external software, EMC2.

Generating a high quality unstructured quad mesh can be a time-consuming task. Let's note that, for wave propagation problems without dynamic faults, if the acceptable accuracy is low (or large computational resources are available to work with a very fine mesh) a structured mesh in which the element faces do not necessarily follow the material interfaces can be generated with the basic built-in meshing capabilities of SEM2D.

3.2 Meshing features included in the solver

The solver itself has very limited meshing capabilities. It can only generate a structured mesh for a single quadrilateral domain, possibly with curved sub-horizontal boundaries and curved sub-horizontal layer interfaces. The domain can be cut in the horizontal direction by a single fault, possibly curved or kinked.

For further details see the Reference Guide for the input blocks `MESH_CART` and `MESH_LAYERED` in Section 4.4.

3.3 Meshing with the MESH2D Matlab utilities

A number of Matlab functions for 2D meshing are provided in `POST/mesh2d`. These can generate structured meshes for quadrilateral domains with curved boundaries, and merge several such meshes to generate a more complicated, globally unstructured mesh. Functions for manipulating, visualizing and exporting these meshes are included. Here is an overview of available tools:

`SEM2DPACK/PRE/MESH2D` provides utilities for the generation, manipulation and visualization of structured 2D quadrilateral meshes, and unstructured compositions of structured meshes.

Mesh generation:

<code>MESH2D_TFI</code>	generates a structured mesh by transfinite interpolation
<code>MESH2D_QUAD</code>	generates a structured mesh for a quadrilateral domain
<code>MESH2D_CIRC_HOLE</code>	generates a mesh for a square domain with a circular hole

MESH2D_WEDGE	generates a mesh for a triangular wedge domain
MESH2D_EXAMPLE1	mesh for a shallow layer over half-space with dipping fault

Mesh manipulation:

MESH2D_ROTATE	rotates the node coordinates
MESH2D_TRANSLATE	translates the node coordinates
MESH2D_MERGE	merges several meshes into a single mesh
MESH2D_WRITE	writes a 2D mesh database file (*.mesh2d)
MESH2D_READ	reads a 2D mesh database from a *.mesh2d file

Mesh visualization:

MESH2D_PLOT	plots a 2D mesh
-------------	-----------------

Miscellaneous tools:

SAMPLE_SEGMENTS	generates points that regularly sample a multi-segment line
-----------------	---

The functions MESH2D_TFI and MESH2D_MERGE are the core tools. The script MESH2D_EXAMPLE1 is a good starting point. The syntax of the mesh database file, *.mesh2d, is described in Section 4.4.

3.4 Generating a mesh with EMC2

3.4.1 The mesh generator EMC2

EMC2 is one of the few public domain 2D mesh generation softwares that includes quadrilateral elements and a Graphical User Interface. Its C code sources and executables can be freely downloaded from

<http://www.ann.jussieu.fr/~hecht/ftp/emc2/>.

We show here an example featuring the most useful functionalities of EMC2. For further details you should refer to the complete documentation of the EMC2 package,

Before starting you must prepare files containing in 2-column data the coordinates (X,Z) of all the points needed to define the geometry of the model (topography, sediment bottom).

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

3.4.2 Notations

The following notations are assumed in the next section:

- (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

3.4.3 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 SPECSEM90. You probably need to get the coordinates of an existing 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 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**

3.4.4 Further 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 existing 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 existing mesh use the script `SEM2DPACK/POST/href.csh`. It edits the `*.emc2.bd` file. You can then restore it in EMC2 and save it in `*.ftq` format.
- To create a fault, in EDIT_MESH mode:
 - a. Crack an existing 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

The solver SEM2D

4.1 About the method

Given a crustal model meshed with quadrilateral elements and a set of material properties, sources, receivers and boundary conditions, SEM2D solves the elastic wave equation applying the Spectral Element Method (SEM) for the space discretization and a second-order explicit scheme for the time discretization. The range of physical problems solved by SEM2D (material constitutive equations and boundary conditions) is described in more detail in Chapter 2. The SEM, introduced by Patera (1984) in Computational Fluid Dynamics, can be seen as a domain decomposition version of the Spectral Method or as a high order version of the Finite Element Method. It inherits from its parent methods the accuracy (spectral convergence), the geometrical flexibility and the natural implementation of mixed boundary conditions.

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 with 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, in Gaetano Festa’s Ph.D. dissertation² and in Kaneko *et al.* (2008).

A more accessible tutorial code, SBIEMLAB written in Matlab, can be downloaded from the author’s website, at web.gps.caltech.edu/~ampuero/software.html.

¹web.gps.caltech.edu/~ampuero/publications.html

²people.na.infn.it/~festa/

4.2 Basic usage flow

In general, a simulation requires the following steps:

1. Prepare the input file `Par.inp` (Section 4.3 and Section 4.4).
2. Run the solver in “check mode” (`iexec=0` in the `GENERAL` input block of `Par.inp`):
`sem2dsolve > info &`.
3. Verify the resolution, stability, estimated CPU cost and memory cost (Section 4.5).
4. If needed go back to step 1 and modify `Par.inp` (Section 4.5), else proceed to next step.
5. Run the solver in “production mode” (`iexec=1`): `sem2dsolve`.
6. Plot and manipulate the solver results (Section 4.6).

Full details are given in the following sections.

4.3 General format of the input file

The input file must be called `Par.inp`. Its typical structure is illustrated by two examples in Figure 4.1 and Figure 4.2. Most of the file is made of standard FORTRAN 90 NAMELIST input blocks. Each block gives input for a specific aspect of the simulation: material properties, sources, receivers, boundary conditions, etc.

The general syntax of a NAMELIST block can be found in any FORTRAN 90 textbook. In summary, a block named `STUFF`, with possible input arguments `a`, `b` and `c`, must be given as

```
&STUFF a=..., b=..., c=... /
```

where ... are user input values. Line breaks and comments preceded by `!` are allowed within an input block.

The complete Reference Guide of the input blocks is presented in Section 4.4. For each block the documentation includes its name, possibly the name of a group of blocks to which it belongs, its purpose, its syntax, the list of its arguments with their description, and some important notes. In the syntax description, a vertical bar `|` between two arguments means “one or the other”. In the argument list, each item is followed by two informations within brackets `[]`. The first bracketed information is the type of the argument: double precision (`double`), integer (`int`), logical (`log`), single character (`char`), fixed length word (e.g. `char*6` is a 6 characters word), arbitrary length word (`name`) or vectors (e.g. `int(2)` is a two element integer vector). The second bracketed information is the default value of the argument. Some arguments are optional, or when absent they are automatically assigned the default values.

Some arguments have a second version with a suffix `H` that allows to set values that are spatially non uniform. The `H`-version of the argument must be set to the name of any of the input blocks of the `DISTRIBUTIONS` group. The appropriate `&DIST_xxxx` block must follow immediately. For example, to set the argument `eta` to a Gaussian distribution:

```
&MAT_KV etaH='GAUSSIAN' /  
&DIST_GAUSSIAN length=1d6,100d0, ampli=0.1d0 /
```

Arguments that accept an H-version are indicated in Section 4.4. When more than one H-version argument is present, the `&DIST_xxxx` blocks must appear in the same order as in the argument list of Section 4.4.

In the next section, Input Block Reference Guide, you should get acquainted with the syntax of the blocks you are most likely to use. The mandatory or more important input blocks are:

- `&GENERAL`
- `&MESH_DEF`, followed by a `&MESH_Method` block
- `&MATERIAL`, followed by a `&MAT_Material` block
- `&BC_DEF`, one for each boundary condition, each followed by a `&BC_Kind` block
- `&TIME`
- `&SRC_DEF`, followed by `&STF_SourceTimeFunction` and `&SRC_Mechanism` blocks
- `&REC_LINE`

Printed by Jean Paul Ampuero

Mar 06, 08 10:48	Par.inp	Page 1/1
<pre> # Parameter file for SEM2DPACK 2.0 #----- Some general parameters ----- &GENERAL iexec=1, ngll= 6, fmax=1.25d0 , ndof=1 , title = 'Test SH', verbose='1111', ItInfo = 1000 / #----- Build the mesh ----- &MESH_DEF method = 'CARTESIAN' / &MESH_CART xlim=0.d0,30.d0 ,zlim=0.d0,30.d0 , nele=60,60/ #---- Elastic material parameters ----- &MATERIAL tag=1, kind='ELAST' / &MAT_ELASTIC rho=1.d0, cp=1.7321d0, cs=1.d0 / #----- Boundary conditions ----- &BC_DEF tag = 2 , kind = 'ABSORB' / &BC_ABSORB stacey=F/ &BC_DEF tag = 3 , kind = 'ABSORB' / &BC_ABSORB stacey=F/ #---- Time scheme settings ----- &TIME TotalTime=35.d0, courant = 0.3d0 / #---- Sources ----- &SRC_DEF stf= 'RICKER' ,coord= 0.d0,0.d0 , mechanism= 'FORCE' / &STF_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 ----- &SNAP_DEF itd=100000, ps=F , bin=F / </pre>		
Thursday March 06, 2008		1/1

Figure 4.1: Input file `Par.inp` for an elementary example in `EXAMPLES/TestSH/` : a boxed region with a structured mesh.

Printed by Jean Paul Ampuero

Mar 06, 08 10:49	Par.inp	Page 1/1
<pre> #----- Some general parameters ----- &GENERAL Iexec =0 , Ngll = 6 , fmax = 1.5 , ndof=1, Title = 'Palos Grandes NS meshed with EMC2' , Verbose='1111', ItInfo = 1000/ #----- Build the mesh ----- &MESH_DEF Method = 'EMC2' / &MESH_EMC2 File= 'NS03qb.ftq' / #----- Elastic material parameters ----- &MATERIAL tag=1, kind='ELAST' / &MAT_ELASTIC rho=1800.d0, cp=850.d0, cs=450.d0/ &MATERIAL tag=2, kind='ELAST' / &MAT_ELASTIC rho=2100.d0, cp=1800.d0, cs=650.d0/ &MATERIAL tag=3, kind='ELAST' / &MAT_ELASTIC rho=2400.d0, cp=2300.d0, cs=850.d0/ &MATERIAL tag=4, kind='ELAST' / &MAT_ELASTIC rho=2600.d0, cp=3800.d0, cs=2200.d0/ #&MAT_ELASTIC rho=2500.d0, cp=5000.d0, cs=2900.d0/ #----- Boundary conditions ----- &BC_DEF Tag = 2, Kind = 'ABSORB' / &BC_ABSORB Stacey=F / &BC_DEF Tag = 3, Kind = 'ABSORB' / &BC_ABSORB Stacey=F, let_wave=T / &BC_DEF Tag = 4, Kind = 'ABSORB' / &BC_ABSORB Stacey=F / #----- Time scheme settings ----- &TIME TotalTime=25.d0, Courant = 0.55d0 / &TIME_NEWMARK alpha=1.d0, beta=0.d0, gamma=0.5d0 / #----- Sources ----- &SRC_DEF stf='RICKER', Mechanism='WAVE' Coord= -1160000.d0,-2000.d0 / &STF_RICKER f0 = 1.d0 , Onset = 1.5d0 ,Ampli = 1.d0 / &SRC_FORCE Angle = 90. / &SRC_WAVE Angle = 30. , phase='S' / #----- 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 = 31 , First = -1163068.0d0,1.d3, Last = -1159697.36d0,1.d3, Isamp=10 / #----- Plots settings ----- &SNAP_DEF itd=100000, fields='V', components='x' / # itd = 3500 &SNAP_PS Mesh=T, Vectors=F, Color=T, Interpol = T, DisplayPts=7, ScaleField=0.2d0 / </pre>		
Thursday March 06, 2008		1/1

Figure 4.2: Input file `Par.inp` for a more realistic example: a sedimentary basin with an unstructured mesh generated by EMC2. Available in `EXAMPLES/UsingEMC2/`.

4.4 Input Blocks Reference Guide

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

```
-----

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

```

stacey  [log] [F] Apply Stacey absorbing conditions for P-SV.
           Higher order than Clayton-Engquist (the default).
let_wave [log] [T] Allow incident waves across this boundary
           if mechanism='WAVE' in &SRC_DEF
```

```
NOTE   : Only implemented for vertical and horizontal boundaries.
```

```
-----

NAME   : BC_DIRNEU
GROUP  : BOUNDARY_CONDITION
PURPOSE: Dirichlet (null displacement)
           and/or Neumann (null or time-dependent traction)
           boundary conditions on vertical or horizontal boundaries
SYNTAX : &BC_DIRNEU h, v, hsrc, vsrc /
           possibly followed by one or two STF_XXXX blocks

h       [char]['N'] Boundary condition on the horizontal component
v       [char]['N'] Boundary condition on the vertical component :
           'N' : Neumann
           'D' : Dirichlet
hsrc    [name]['null'] Name of the source time function for a
           time-dependent horizontal traction:
           'RICKER', 'TAB', 'USER', etc (see STF_XXXX input blocks)
vsrc    [name]['null'] Same for the vertical component
```

```
-----

NAME   : BC_DYNFLT
GROUP  : BOUNDARY_CONDITION, DYNAMIC_FAULT
PURPOSE: Dynamic fault with friction
SYNTAX : &BC_DYNFLT friction, opening, Tn|TnH, Tt|TtH,
```

Sxx|SxxH, Sxy|SxyH, Sxz|SxzH, Syz|SyzH, Szz|SzzH
 ot1, otd, oxi, osides /

followed, in order, by:

1. &DIST_XXX blocks (from the DISTRIBUTIONS group) for arguments with suffix H, if present, in the order listed above.
2. &BC_DYNFLT_SWF, &BC_DYNFLT_TWF or &BC_DYNFLT_RSF block(s) (if absent, default values are used)
3. &BC_DYNFLT_NOR block (if absent, default values are used)

friction [name(2)] ['SWF',''] Friction law type:

SWF = slip weakening friction

TWF = time weakening friction

RSF = rate and state dependent friction

Some friction types can be combined. E.g. to set the friction coefficient to the minimum of SWF and TWF, set
 friction='SWF','TWF'

opening	[log]	[T]	Allow fault opening instead of tensile normal stress
Tn	[dble]	[0d0]	Initial normal traction (positive = tensile)
Tt	[dble]	[0d0]	Initial tangent traction (positive antiplane: y>0)
Sxx	[dble]	[0d0]	Initial stress sigma_xx
Sxy	[dble]	[0d0]	Initial stress sigma_xy
Sxz	[dble]	[0d0]	Initial stress sigma_xz
Syz	[dble]	[0d0]	Initial stress sigma_yz
Szz	[dble]	[0d0]	Initial stress sigma_zz
otd	[dble]	[0.d0]	Time lag between outputs (in seconds) Internally adjusted to the nearest multiple of the timestep Its value can be found in the output file FltXX_sem2d.hdr The default internally resets otd = timestep
ot1	[dble]	[0.d0]	Time of first output (in seconds) Internally adjusted to the nearest multiple of the timestep Its value can be found in the output file FltXX_sem2d.hdr
oxi	[int(3)]	[(1,huge,1)]	First, last node and stride for output The default resets oxi(2) = last fault node
osides	[log]	[F]	Export displacement and velocities on each side of the fault

NOTE: the initial stress can be set as a stress tensor (Sxx,etc), as
 initial tractions on the fault plane (Tn and Tt) or as the sum of both.

NOTE: we recommend to use dynamic faults with the leapfrog time scheme
 and a layer of Kelvin-Voigt damping material near the fault.

NAME : BC_DYNFLT_NOR

GROUP : DYNAMIC_FAULT

PURPOSE: Normal stress response for dynamic faults.

SYNTAX : &BC_DYNFLT_NOR kind, V, L, T /

kind [int] [1] Type of normal stress response:
 0 = shear strength is independent of normal stress
 (the cohesive strength is set as the product of
 friction coefficient and initial normal stress)
 1 = Coulomb
 2 = Prakash-Clifton with regularizing time scale
 3 = Prakash-Clifton with regularizing length scale
 T [dble] [1d0] Regularization time scale if kind=2
 V [dble] [1d0] Characteristic velocity if kind=3
 L [dble] [1d0] Regularization length scale if kind=3

NAME : BC_DYNFLT_RSF

GROUP : DYNAMIC_FAULT

PURPOSE: Velocity and state dependent friction

SYNTAX : &BC_DYNFLT_RSF kind, Dc | DcH, Mus | MusH ,
 a | aH, b | bH, Vstar | VstarH /
 followed by &DIST_XXX blocks (from the DISTRIBUTIONS group) for
 arguments with suffix H, if present, in the order listed above.

kind [int] [1] Type of rate-and-state friction law:
 1 = strong velocity-weakening at high speed
 as in Ampuero and Ben-Zion (2008)
 Dc [dble] [0.5d0] Critical slip
 MuS [dble] [0.6d0] Static friction coefficient
 a [dble] [0.01d0] Direct effect coefficient
 b [dble] [0.02d0] Evolution effect coefficient
 Vstar [dble] [1d0] Characteristic or reference slip velocity

NAME : BC_DYNFLT_SWF

GROUP : DYNAMIC_FAULT

PURPOSE: Slip-weakening friction

SYNTAX : &BC_DYNFLT_SWF Dc | DcH, MuS | MuSH , MuD | MuDH, healing /
 followed by &DIST_XXX blocks (from the DISTRIBUTIONS group) for
 arguments with suffix H, if present, in the order listed above.

kind [int] [1] Type of slip weakening function:
 1 = linear
 2 = exponential
 Dc [dble] [0.5d0] Critical slip
 MuS [dble] [0.6d0] Static friction coefficient
 MuD [dble] [0.5d0] Dynamic friction coefficient

healing [log] [F] Instantaneous healing upon slip arrest
 Healing is currently valid only with the leapfrog time scheme

NAME : BC_DYNFLT_TWF
 GROUP : DYNAMIC_FAULT
 PURPOSE: Time weakening friction for dynamic faults
 with prescribed rupture speed.
 SYNTAX : &BC_DYNFLT_TWF MuS, MuD, Mu0, X, Z, V, L, T /

MuS [dble] [0.6d0] Static friction coefficient
 MuD [dble] [0.5d0] Dynamic friction coefficient
 Mu0 [dble] [0.6d0] Friction coefficient at the hypocenter at time=0
 X,Z [dble] [0d0] Position of hypocenter
 V [dble] [1d3] Rupture propagation speed
 L [dble] [1d0] Size of weakening zone
 T [dble] [huge] Total duration

NAME : BC_DEF
 PURPOSE: Define a boundary condition
 SYNTAX : &BC_DEF tag, tags, kind /
 possibly followed by &BC_kind 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:
 'DIRNEU', 'ABSORB', 'PERIOD', 'LISFLT', 'DYNFLT'

NOTE : Most of the boundary conditions need additional data, given
 in a BC_kind input block of the BOUNDARY_CONDITIONS group
 immediately following the BC_DEF block.

NAME : BC_LSF

GROUP : BOUNDARY_CONDITION

PURPOSE: Linear slip fault, a displacement discontinuity interface
where stress and slip are linearly related

SYNTAX : &BC_LSF Ktang | Ctang, Knorm | Cnorm /

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_

If C=0d0 or K=Inf then no discontinuity is allowed (transparent)

If K=0d0 the fault is free stress boundary

NAME : DIST_GAUSSIAN

GROUP : DISTRIBUTIONS_2D

PURPOSE: Bell shaped Gaussian 2D distribution

SYNTAX : &DIST_GAUSSIAN centered_at, length, offset, ampli, order /

centered_at	[dble(2)] [0,0]	Coordinates of the center point.
length	[dble(2)] [1]	Characteristic lengths on each axis.
offset	[dble] [0]	Background level.
ampli	[dble] [1]	Amplitude from background.
order	[int] [1]	Exponent

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 : Make sure the 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*double] bottom-left corner

dx,dz = [2*double] spacing along x and z

Line 2 to nx*nz+1 : [ncol*double] values at grid points

listed from left to right (x0 to x0+nx*dx),

then from bottom to top (z0 to z0+nz*dz)

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 or Z.

SYNTAX : &DIST_LINEAR n,dim,length /

followed immediately by the interpolation data,

one line per point, two columns: position (X or Z), value

Position must be sorted in increasing order.

or

&DIST_LINEAR file,dim,length /

and the interpolation data is read from a two-column file

n [int] [0] Number of points to be interpolated

dim [int] [1] Interpolate along X (dim=1) or along Z (dim=2)
file [name] [none] Name of the ASCII file containing the data
length [dble] [0] Smoothing length for sliding average window
 No smoothing if length=0

NAME : DIST_ORDERO
GROUP : DISTRIBUTIONS_2D
PURPOSE: Blockwise constant 2D distribution.

SYNTAX : &DIST_ORDERO 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 $R \leq r(1)$,
 second $r(1) < R \leq r(2)$, ...
 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 or Z.
SYNTAX : &DIST_SPLINE file,dim /

file [name] [none] Name of the ASCII file containing
the interpolation data, one line per point, two columns:
one line per point, two columns: position (X or Z), value
Position must be sorted in increasing order.

dim [int] [1] Interpolate along X (dim=1) or along Z (dim=2)

NAME : GENERAL

PURPOSE: General parameters

SYNTAX : &GENERAL iexec, ngll, fmax, title, verbose, itInfo /

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 issue a warning if not. 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

title [word] [none] Title of the simulation

verbose [char(4)] ['1101'] Print progress information during each phase:
verbose(1) = input phase
verbose(2) = initialization phase
verbose(3) = check phase
verbose(4) = solver phase
Example: '0001' is verbose only during solver.

itInfo [int] [100] Frequency (in number of timesteps) for printing
progress information during the solver phase.

NAME : MAT_DAMAGE

GROUP : MATERIALS

PURPOSE: Set material properties for the damage rheology of
Lyakhovsky, Ben-Zion and Agnon (J. Geophys. Res. 1997)
and Hamiel et al (Geophys. J. Int. 2004)

SYNTAX : &MAT_DAMAGE cp,cs,rho,phi,alpha,Cd,R,e0,ep /

cp [dble] [0d0] P wave velocity

```

cs      [dble][0d0] S wave velocity
rho     [dble][0d0] density
phi     [dble][0d0] internal friction angle
alpha   [dble][0d0] initial value of damage variable
Cd      [dble][0d0] damage evolution coefficient
R       [dble][0d0] damage-related plasticity coefficient Cv
           normalized by the inverse of the intact shear modulus
e0      [dble(3)][0d0] initial total strain (11, 22 and 12)
ep      [dble(3)][0d0] initial plastic strain (11, 22 and 12)

```

```

NAME    : MAT_ELASTIC
GROUP   : MATERIALS
PURPOSE: Set material properties for a linear elastic medium
SYNTAX  : For isotropic material:
           &MAT_ELASTIC rho|rhoH, cp|cpH, cs|csH /
           followed by DIST_XXXX blocks, for arguments with suffix H,
           if present, in the same order as listed above.
           For transverse anisotropy with vertical symmetry axis:
           &MAT_ELASTIC rho, c11,c13,c33,c55 /

cp      [dble][0d0] P wave velocity
cs      [dble][0d0] S wave velocity
rho     [dble][0d0] density
c11,c13,c33,c55 [dble][0d0] anisotropic elastic moduli

```

```

NAME    : MATERIAL
PURPOSE: Define the material type of a tagged domain
SYNTAX  : &MATERIAL tag, kind /
           followed by one or two MAT_XXXX input blocks.

tag     [int] [none]   Number identifying a mesh domain
kind    [name(2)] ['ELAST',''] Material types:
           'ELAST', 'DMG','PLAST', 'KV'

NOTE    : Some combinations of material kinds can be assigned to the same domain.
           Any material type can be combined with 'KV', for instance:
           &MATERIAL tag=1, kind='ELAST','KV' /
           followed by a &MAT_ELAST block and a &MAT_KV block
           sets an elastic material with Kelvin-Voigt damping.

```

```

NAME    : MAT_KV

```

GROUP : MATERIALS

PURPOSE: Sets material properties for a Kelvin-Voigt viscous material.
Adds a damping term $C*v = K*eta*v$, where eta is a viscous time.
This produces attenuation with frequency-dependent quality factor

$$Q(f) = 1/(eta*2*pi*f)$$

Its main usage is for artificial damping of high-frequency numerical artifacts generated by dynamic faults, which requires a thin layer of Kelvin-Voigt elements surrounding the fault with $eta/dt = 0.1$ to 0.3 and a layer thickness of 1 to 2 elements on each side of the fault.

SYNTAX : &MAT_KV eta, ETaxDT /

&MAT_KV etaH, ETaxDT / followed by a DIST_XXX input block

eta [dble][0d0] Viscosity coefficient

ETaxDT [log][T] If eta is given in units of dt (timestep)

NOTE : Kelvin-Voigt viscosity modifies the stability of time integration.

The timestep (or the Courant number) must be set to a value smaller than usual. The critical timestep for a Kelvin-Voigt material integrated with the leapfrog time scheme is

$$dtc_{kv} = eta * (\sqrt{1+dtc^2/eta^2} - 1)$$

where dtc is the critical timestep for a purely elastic medium ($eta=0$).

In terms of the normalized viscosity (if $ETaxDT=T$):

$$dtc_{kv} = dtc / \sqrt{1 + 2*eta}$$

NAME : MAT_PLASTIC

GROUP : MATERIALS

PURPOSE: Set material properties for elasto-plastic material
with Mohr-Coulomb yield criterion
and non-dilatant (null volumetric plastic strain)

SYNTAX : &MAT_PLASTIC cp,cs,rho,phi,coh,Tv,e0 /

cp [dble][0d0] P wave velocity

cs [dble][0d0] S wave velocity

rho [dble][0d0] density

phi [dble][0d0] internal friction angle

coh [dble][0d0] cohesion

Tv [dble][0d0] visco-plastic relaxation time

e0 [dble(3)][0d0] initial total strain (11, 22 and 12)

NAME : MESH_CART

GROUP : MESH_DEF

PURPOSE: Rectangular box with structured mesh.

SYNTAX : &MESH_CART xlim, zlim, nelem, ezflt,fztag, FaultX /

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
 ezflt [int][0] introduce a horizontal fault between the ezflt-th
 and the (ezflt+1)-th element rows. Rows are numbered from
 bottom to top, starting at ezflt=1.
 If ezflt=0, no fault is introduced (default).
 If ezflt=-1, a fault is introduced at the middle of the box
 or near below the middle (ezflt is reset to int[nelem(2)/2])
 fztag [int][0] fault zone tag for elements touching the fault
 Useful to set a damping layer near the fault.
 If ezflt=0, a fault is assumed at the bottom boundary
 FaultX [log] [F] Same as ezflt=-1. Obsolete (will be deprecated)

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] Horizontal index of the first and last elements.
 The leftmost element column has horizontal index 1.
 ez [int(2)] [none] Vertical index of the first and last elements.
 The bottom element row has vertical index 1.

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 /
 followed by a &MESH_method input block

method [name] [none] Meshing method name:
 'CARTESIAN', 'LAYERED', 'EMC2', 'MESH2D'

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

xlim [dble(2)] [none] X limits of the box (min and max)
 zmin [dble] [none] bottom Z limit of the box
 nx [int] [1] Number of elements along the X direction.
 Not needed if ztopH='QSPLINE' in a &MESH_LAYER block.
 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.
 ezflt [int] [0] introduce a fault between the ezflt-th and the
 (ezflt+1)-th element rows, numbered from bottom to top.
 If ezflt=0 (default), no fault is introduced.
 fztag [int] [0] fault zone tag for elements touching the fault
 Useful to set a damping layer near the fault.

NOTE: the following tags are automatically assigned to the boundaries:

- 1 Bottom
- 2 Right
- 3 Top
- 4 Left
- 5 Fault, lower side

6 Fault, upper side

NAME : MESH_LAYER
GROUP : MESH_DEF
PURPOSE: Define mesh parameters for one layer
SYNTAX : &MESH_LAYER nz, ztop|ztopH, tag /
 followed by a DIST_XXXX block if ztopH is set

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'] Name of the type of spatial distribution to
 generate an irregular (non flat) top boundary. In general it is
 one of the 1D distribution available through a DIST_XXXX block:
 ztopH = 'LINEAR', or
 ztopH = 'SPLINE', etc.
 To generate a curve with a smooth normal, typically to guarantee
 smooth boundary conditions on curved faults, set:
 ztopH='QSPLINE', followed by a &QSPLINE 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)

NOTE: If ztopH='LINEAR' the mesh uses linearly deformed (Q4) elements,
 otherwise it uses quadratically deformed (Q9) elements

NAME : QSPLINE
GROUP : MESH_LAYER
PURPOSE: Define the boundary of a layer using quadratic splines and
 enforcing smooth (continuous) normal between elements, for instance
 to guarantee smooth boundary conditions on curved faults.

SYNTAX : &QSPLINE file /

file [string] [''] Name of ASCII file containing information of
 all the element vertex nodes lying on the boundary curve.
 One line per node, ordered by increasing x, 3 columns per line:
 (1) x position
 (2) z position
 (3) derivative dz/dx of the curve at the node
 All QSPLINE curves in a mesh must have the same number of nodes.
 The parameter nx in &MESH_LAYERED is automatically reset
 (nx = number of nodes in QSPLINE - 1)

NAME : MESH_MESH2D
GROUP : MESH_DEF
PURPOSE: Imports a mesh in mesh2d format
as defined by the PRE/mesh2d mesh generator tools for Matlab
SYNTAX : &MESH_MESH2D file /

file [name] [none] Name of the MESH2D file, including suffix.
The format of this file is:

"NEL NPEL NNOD NBC"
1 line with 4 integers:
nb of elements, nodes per element, total nb of nodes, nb of boundaries
"NID X Y"
NNOD lines, one per node, with 1 integer and 2 reals:
node id, x, y
"EID NODES TAG"
NEL lines, one per element, with NPEL+2 integers:
element id, NPEL node ids, tag.
"BCTAG NBEL" |
2 integers: boundary tag, nb of boundary elements |
"BID EID EDGE" | repeat for each of
NBEL lines, one per boundary element, 3 integers: | the NBC boundaries
boundary element id, bulk element id, edge id |

NAME : SNAP_DEF
GROUP : SNAPSHOT_OUTPUTS
PURPOSE: Set preferences for exporting snapshots
SYNTAX : &SNAP_DEF it1, itd, fields, components, bin, visual3, avs, ps, gmt /
Followed by a &SNAP_PS block if ps=T.

it1 [int] [0] Time step of first snapshot output
itd [int] [100] Number of timesteps between snapshots
fields [char*] ['V'] fields to export in snapshots (the prefix of the
output file names is 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)
'd' divergence rate (dvx/dx + dvz/dz)
'c' curl rate (dvx/dz - dvz/dx)
components [char*] ['ya'] components for PostScript outputs:

in P-SV: 'x','z' and/or 'a' (amplitude). 'y' is ignored
 in SH: 'y' only. Other values are ignored.

ps [log] [T] PostScript (see &SNAP_PS input block)
 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 (only for D,V and A fields)
 visual3 [log] [F] Visual3 (only for D,V and A fields)
 bin [log] [T] binary

NOTE : E and S fields are exported only as binary.

NAME : SNAP_PS

GROUP : SNAPSHOT_OUTPUTS

PURPOSE: Preferences for PostScript snapshots

SYNTAX : &SNAP_PS 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] [F] 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 : If single receiver line:

&REC_LINE number,first,last,AtNode,isamp,field,irepr /

If receiver locations from file:

&REC_LINE file,AtNode,isamp,field,irepr /

number	[int] [0]	Number of stations in the line
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: X and Z (in meters)
AtNode	[log] [T]	Relocate the stations at the nearest GLL node
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
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 : SOURCE MECHANISM

PURPOSE: Point force source

SYNTAX : &SRC_FORCE angle /

angle	[dble] [0d0]	For P-SV, the angle of the applied force, in degrees, counterclockwise from Z-UP, e.g.: 90 points left, 180 points down For SH, angle is ignored and the SRC_FORCE block is not required.
-------	--------------	--

NAME : SRC_DEF

PURPOSE: Define the sources.

SYNTAX : &SRC_DEF stf, mechanism, coord /

&SRC_DEF stf, mechanism, file /

followed by one SOURCE TIME FUNCTION block (STF_XXXX)
and one SOURCE MECHANISM block (SRC_XXXX)

```

stf      [name] [none] Name of the source time function:
          'RICKER', 'TAB', 'HARMONIC', 'BRUNE' or 'USER'
mechanism [name] [none] Name of the source mechanism:
          'FORCE', 'EXPLOSION', 'DOUBLE_COUPLE', 'MOMENT' or 'WAVE'
coord     [dble(2)] [huge] Location (x,z) of the source (m).
file      [name] ['none'] Name of file containing source parameters.
          The file format is ASCII with one line per source and
          2, 3 or 4 columns per line:
              (1) X position (in m)
              (2) Z position (in m)
              (3) time delay (in seconds)
              (4) relative amplitude
          If column 4 is absent, amplitude = 1.
          If columns 3 and 4 are absent, delay = 0 and amplitude = 1.

```

```

-----
NAME      : SRC_DOUBLE_COUPLE
GROUP     : SOURCE 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.

NOTE : The source time function gives the cumulative seismic moment $M_o(t)$, NOT the seismic moment rate.

NOTE : The seismic moment M_o must be rescaled because a 2D point source is equivalent to a 3D line source. A proper scaling is obtained by dividing the original 3D moment by the characteristic size of the rupture area in the off-plane dimension. An approximate scaling for a fault area with aspect ratio close to unity is

$$M_{o_2D} = (M_{o_3D}/d\tau)^{2/3} * d\tau$$

where $d\tau$ is the stress drop (typically a few MPa).

```

-----
NAME      : SRC_MOMENT
GROUP     : SOURCE MECHANISM

```

PURPOSE: Define a moment tensor source

SYNTAX : &SRC_MOMENT Mxx,Mxz,Mzx,Mzz /
 &SRC_MOMENT Myx,Myz /

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

NAME : SRC_WAVE

GROUP : SOURCE 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_BRUNE

GROUP : SOURCE TIME FUNCTIONS

PURPOSE: Brune (1970)'s model with omega-squared spectral fall-off:

stf(t) = ampli*(1 - (1+2*pi*fc*t)*exp(-2*pi*fc*t))

SYNTAX : &STF_BRUNE ampli, fc /

ampli [dble] [1d0] Amplitude (usually the seismic moment)
 fc [dble] [1d0] Corner frequency (Hz)

NAME : STF_HARMONIC

GROUP : SOURCE TIME FUNCTIONS

PURPOSE: Harmonic source time function $f(t) = \text{ampli} \cdot \sin(2\pi \cdot t \cdot f_0)$

SYNTAX : &STF_HARMONIC ampli, f0 /

ampli [dble] [0d0] Amplitude

f0 [dble] [0d0] Frequency

NAME : STF_RICKER

GROUP : SOURCE TIME FUNCTIONS

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 $f_{\max} = 2.5 \times f_0$.

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 $2.5 \times f_0$ there is little energy, this is a
 recommended value for f_{\max} .

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 : SOURCE TIME FUNCTIONS

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: time and value. Time can be irregularly sampled and
 must increase monotonically.

NOTE : assumes $\text{value}(t < \min(\text{time})) = \text{value}(\min(\text{time}))$
 and $\text{value}(t > \max(\text{time})) = \text{value}(\max(\text{time}))$

NAME : STF_USER

GROUP : SOURCE TIME FUNCTIONS

PURPOSE: A template for user-supplied source time function.

File stf_user.f90 must be modified by the user to fit
 special needs.

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

```

NAME : TIME

PURPOSE: Defines time integration scheme

SYNTAX : &TIME kind, {Dt or Courant}, {NbSteps or TotalTime} /
Possibly followed by a TIME_XXXX block.

```

kind      [char*10] ['leapfrog'] Type of scheme:
          'newmark'      Explicit Newmark
          'HHT-alpha'    Explicit HHT-alpha
          'leapfrog'     Central difference
          'symp_PV'      Position Verlet
          'symp_PFR'     Position Forest-Ruth (4th order)
          'symp_PEFRL'   Extended PFR (4th order)
Dt         [dble] [none] Timestep (in seconds)
Courant    [dble] [0.5d0] the maximum value of the Courant-Friedrichs-Lewy
                    stability number (CFL), defined as
                    CFL = Dt*wave_velocity/dx
                    where dx is the distance between GLL nodes. Typically CFL<= 0.5
NbSteps    [int] [none] Total number of timesteps
TotalTime  [int] [none] Total duration (in seconds)

```

NOTE : The leap-frog scheme is recommended for dynamic faults. It is equivalent to the default Newmark scheme (beta=0, gamma=1/2). However it is faster and requires less memory.

NAME : TIME_NEWMARK

GROUP : TIME SCHEMES

PURPOSE: Explicit Newmark time integration scheme

SYNTAX : &TIME_NEWMARK gamma, beta /

```

beta      [dble] [0d0] First Newmark parameter.
          If beta=0 the scheme is fully explicit (the update of
                    displacement depends only on the last value of acceleration),
                    otherwise it is a single-predictor-corrector scheme
gamma     [dble] [0.5d0] Second Newmark parameter.
          Second order requires gamma=1/2.

```

NAME : TIME_HHTA
 GROUP : TIME SCHEMES
 PURPOSE: Explicit HHT-alpha time integration scheme, second order
 SYNTAX : &TIME_HHTA alpha, rho /

alpha [dble] [0.5d0] Parameter in the HHT-alpha method. Values in [0,1].
 Defined here as $1 + \text{HHT's original definition of alpha}$.
 When alpha=1 it reduces to second order explicit Newmark
 (beta=0, gamma=0.5).
 rho [dble] [0.5d0] Minimum damping factor for high frequencies.
 Values in [0.5,1]. Rho=1 is non-dissipative.

NOTE: We consider only second order schemes, for which $\alpha + \gamma = 3/2$
 If $\alpha < 1$, Newmark's beta is related to the HHT parameters by

$$\beta = 1 - \alpha - \rho^2 * (\rho - 1) / [(1 - \alpha) * (1 + \rho)^3]$$

 If $\alpha = 1$, we set $\rho = 1$ ($\beta = 0$, $\gamma = 0.5$)

NOTE: Dissipative schemes ($\rho < 1$) require slightly smaller Courant number
 (0.56 for $\rho = 0.5$, compared to 0.6 for $\rho = 1$)

NOTE: This is an explicit version of the HHT-alpha scheme of
 H.M. Hilber, T.J.R. Hughes and R.L. Taylor (1977) "Improved numerical
 dissipation for time integration algorithms in structural dynamics"
 Earthquake Engineering and Structural Dynamics, 5, 283-292
 implemented with a slightly different definition of alpha (1+original).
 Its properties can be derived from the EG-alpha scheme of
 G.M. Hulbert and J. Chung (1996) "Explicit time integration
 algorithms for structural dynamics with optimal numerical dissipation"
 Comp. Methods Appl. Mech. Engrg. 137, 175-188
 by setting $\alpha_m = 0$ and $\alpha = 1 - \alpha_f$.

4.5 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 `sem2dsolve > 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, and much smaller for highly deformed meshes (see Section 6.1 on “Instabilities in very distorted elements”). This number is defined at each computational node as

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

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 ($\approx \text{Ng11}^2$ times smaller) because SEM internally subdivides each element onto a non-regular grid of $\text{Ng11} \times \text{Ng11}$ 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 $p = \text{Ng11} - 1$, 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 `Ng11` 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 and 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`.

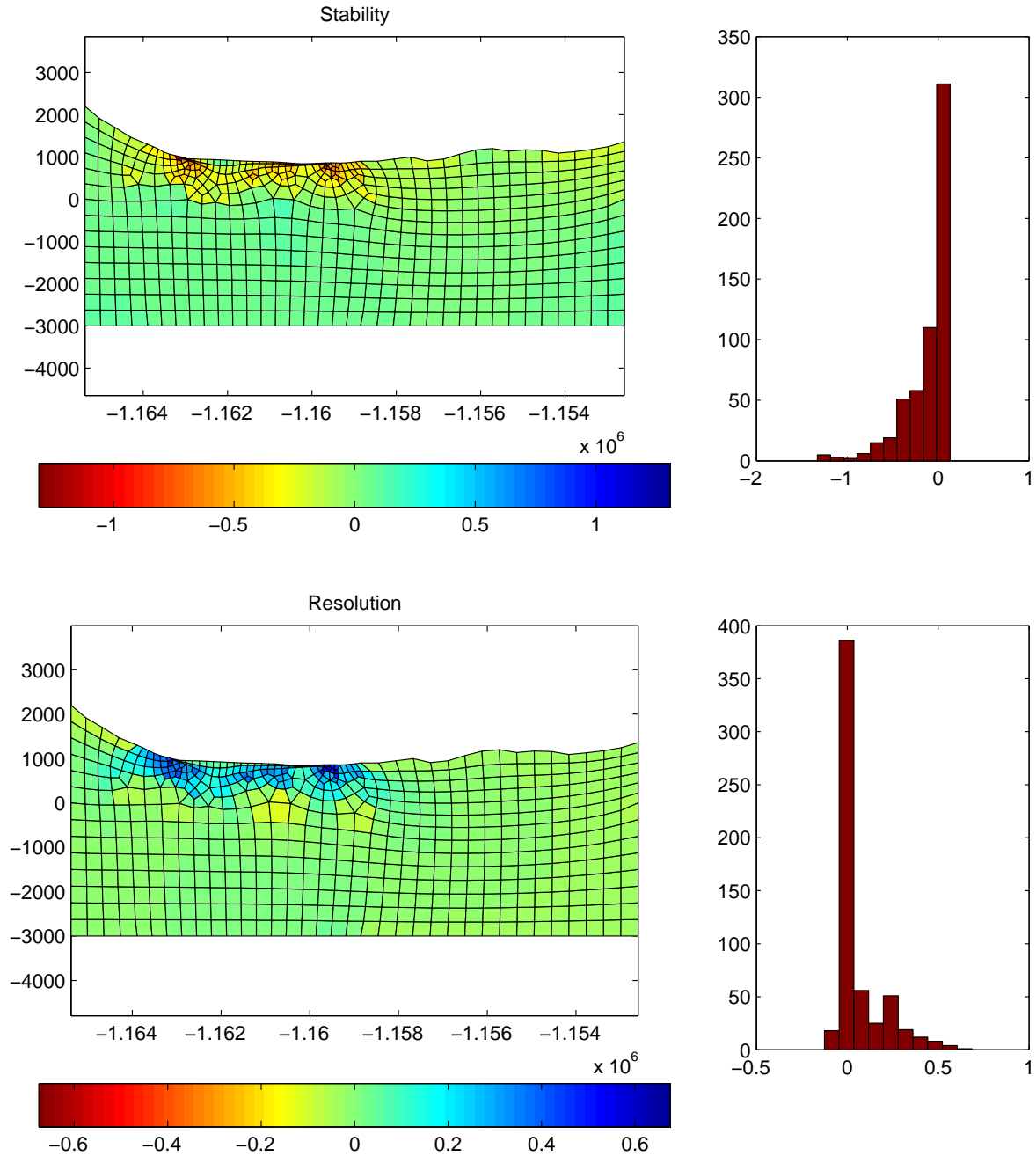


Figure 4.3: Checking the quality of a mesh with `PRE/ViewMeshQuality.m` for the example in `EXAMPLES/UsingEMC2/`. The balance of the stability and resolution properties of the mesh can be analyzed: logarithmic stability index (top) and logarithmic resolution index (bottom). Histograms of these indices (in number of elements) are shown on the right.

The quality of the mesh can be inspected with the Matlab script `PRE/ViewMeshQuality.m` which produces plots like Figure 4.3. The proper balance of the mesh with respect to the following two criteria can be analyzed:

- **Stability criterion**, related to the largest stable timestep. The stability of each element is quantified by

$$S = \min(\Delta x / c_P).$$

We also define a stability index as

$$SI = \log[S / \text{median}(S)].$$

where the median value is taken over the whole mesh. Red elements (small SI) 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. The resolution of each element is quantified by

$$R = \min(c_S / h).$$

We also define a resolution index as

$$RI = \log[R / \text{median}(R)].$$

where the median value is taken over the whole mesh. Red elements (small RI) 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 RI (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 4.3 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.

³In 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.

Printed by Jean Paul Ampuero

```

Mar 06, 08 18:04                               info                               Page 1/4

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

Date : 06 - 03 - 2008                           Time : 18:03:23
-----

*****
*           Input phase                         *
*****

General Parameters
=====
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
Print progress information during
    input phase . . . . . (verbose(1)) = T
    initialization phase . . . . . (verbose(2)) = T
    checking phase . . . . . (verbose(3)) = T
    solver phase . . . . . (verbose(4)) = T
Frequency for solver progress information . (itInfo) = 1000

Mesh Generation
=====
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

Time integration
=====
Scheme . . . . . (kind) = leapfrog
Number of time steps . . . . . (NbSteps) = will be set later
Time step increment . . . . . (Dt) = will be set later
Courant number . . . . . (Courant) = 0.30
Total simulation duration . (TotalTime) = 35.000E+00

Material Properties
=====
Number of materials . . . . . = 1

Material index . . . . . (tag) = 1
Material type . . . . . (kind) = Elastic
P-wave velocity . . . . . (cp) = 1.732E+00
S-wave velocity . . . . . (cs) = 1.000E+00
Mass density . . . . . (rho) = 1.000E+00

```

Thursday March 06, 2008

```

Mar 06, 08 18:04                               info                               Page 2/4

Poisson's ratio . . . . . = 250.021E-03
First Lamé parameter Lambda . . . . . = 1.000E+00
Second Lamé parameter Mu . . . . . = 1.000E+00
Bulk modulus K . . . . . = 1.667E+00
Young's modulus E . . . . . = 2.500E+00

Boundary Conditions
=====

Boundary tag . . . . . (tag) = 2
Boundary condition . . . . . (kind) = ABSORB
Type of absorbing boundary . . . (stacey) = Clayton-Engquist
Allow incident wave . . . . . (let_wave) = T

Boundary tag . . . . . (tag) = 3
Boundary condition . . . . . (kind) = ABSORB
Type of absorbing boundary . . . (stacey) = Clayton-Engquist
Allow incident wave . . . . . (let_wave) = T

Sources
=====
X-position (meters) . . . . (coord(1)) = 0.000E+00
Y-position (meters) . . . . (coord(2)) = 0.000E+00
Source time function . . . . . = Ricker
Fundamental frequency (Hz) . . . . (f0) = 500.000E-03
Time delay (s) . . . . . (onset) = 3.000E+00
Multiplying factor . . . . . (ampli) = 250.000E-03
Source Type . . . . . = Collocated Force
If P-SV: counterclockwise angle / up . = 0.00

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

Snapshot Outputs
=====
Timestep of first snapshot output . . . . . (it1) = 0
Number of timesteps between snapshots . . . . (itd) = 100000
Save results in PS file or not . . . . . (ps) = F
Save grid triangulation for GMT . . . . . (gmt) = 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

*****
* Initialization phase *
*****

```

info

1/2

Printed by Jean Paul Ampuero

```

Mar 06, 08 18:04                               info                               Page 3/4

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 model ..... [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
              for maximum frequency   = 1.250E+00 Hz
              minimum wavelength      = 1.600E+00 m

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

Defining work arrays for elasticity ..... [OK]
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

```

Thursday March 06, 2008

```

Mar 06, 08 18:04                               info                               Page 4/4

5  20.000E+00    0.000E+00    20.000E+00    0.000E+00    0.000E+00
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 = 0.000E+00  vmax = 0.000E+00  dmax = 0.000E+00

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

--- CPU TIME ESTIMATES (in seconds) :
CPU time for initialization . . 806.877E-03
CPU time per timestep . . . . 18.997E-03
Total solver CPU time . . . . 37.747E+00
(mins) . . . . 629.118E-03
(hours) . . . . 10.485E-03

Timestep #    1000  t = 17.621E+00  vmax = 91.661E-03  dmax = 28.653E-03

--- CPU TIME INFORMATION (in seconds) :
CPU time for initialization . . 806.877E-03
CPU time per timestep . . . . 20.395E-03
Total solver CPU time . . . . 40.524E+00
(mins) . . . . 675.397E-03
(hours) . . . . 11.257E-03

Storing sismos data (SEP format) ...

-----
Program  S P E C F E M :  end
-----

Test SH

-----
Date : 06 - 03 - 2008                               Time : 18:04:11
-----

```

info

2/2

4.6 Outputs, their visualization and manipulation

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*`.

4.6.1 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`.

4.6.2 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.

4.6.3 Snapshots

`sem2dsolve` generates snapshots at a constant interval defined, in number of solver timesteps, by the input parameter `itd` of the `SNAP_DEF` input block. An example is shown in Figure 4.4. 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 `s` function `POST/sem2d_snapshot_read.m` to find more about the data formats and their manipulation.

Snapshots can also be exported as PostScript files `xx_XXX_sem2d.ps`. These can be merged into an animated GIF (movie) file `movie.gif` by the script `POST/movie.csh` and displayed by `xanim movie.gif` or `animate movie.gif`. An animated GIF can also be created by the Matlab function `POST/sem2d_snapshot_movie.m`.

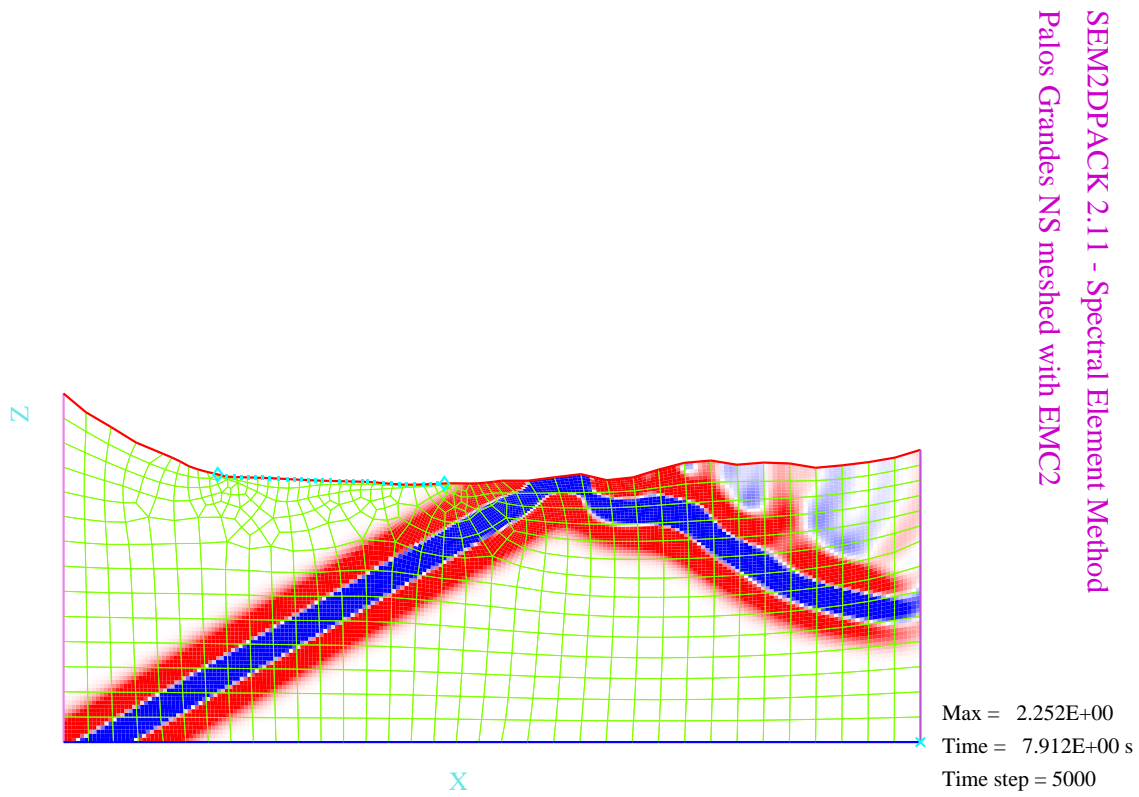


Figure 4.4: Sample snapshot from `EXAMPLES/UsingEMC2/`: an obliquely incident SH plane wave impinging on a sedimentary basin. The unstructured mesh of spectral elements is plotted on background.

4.6.4 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 $NSAMP \times NSTA$ 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 XSU-Seismic Unix visualization tool⁴:

- `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 4.5).

The script `POST/sample_seis.m` shows how to manipulate and plot seismogram data in Matlab. It uses the functions `POST/sem2d_read_seis.m` and `POST/plot_seis.m`.

4.6.5 Fault outputs

Fault data from dynamic rupture simulations is stored in three files (where `XX` is the boundary tag of the first side of the fault, `tags(1)` of the `BC_SWFFLT` input block):

- `F1tXX_sem2d.hdr` contains the information needed to read the other fault data files. Its format, line by line, is:
 1. `NPTS NDAT NSAMP DELT` (name of parameters)
 2. Value of parameters above
 3. Name of fields exported in `F1tXX_sem2d.dat`, 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

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

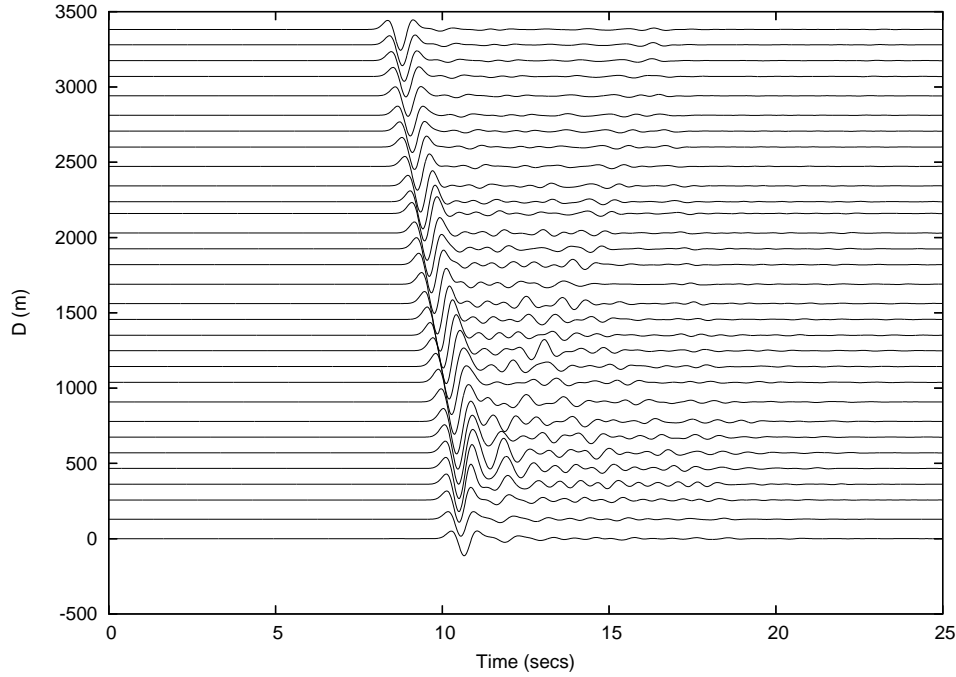


Figure 4.5: Sample seismograms from `EXAMPLES/UsingEMC2/` generated with `POST/seis_plot.csh`.

- `FltXX_sem2d.dat` contains the space-time distribution of fault data such as slip, slip rate, stress and strength. Every `DELT` seconds a block of fault data values is written. The total number of blocks is `NSAMP`. Each block has `NDAT` lines (one per fault data field) and `NPTS` columns (one per fault node)⁵. Stresses are relative to their initial values.
- `FltXX_init_sem2d.tab` contains the spatial distribution of initial shear stress, initial normal stress and initial friction (3 columns).
- `FltXX_potency_sem2d.tab` contains time-series of seismic potency and potency rate. The seismic potency tensor p_{ij} is defined by the following integral along the fault:

$$p_{ij} = \frac{1}{2} \int_{fault} (n_i \Delta u_j + n_j \Delta u_i) dx \quad (4.1)$$

where Δu is slip and n is the local unit vector normal to the fault. The file contains one line per timestep. In SH (`ndof=1`) each line has 4 columns: 2 components of potency (p_{13} and p_{23}) and 2 components of potency rate (\dot{p}_{13} and \dot{p}_{23}). In P-SV (`ndof=2`) each line has 3 components of potency (p_{11} , p_{22} and p_{12}) and 3 components of potency rate (\dot{p}_{11} , \dot{p}_{22} and \dot{p}_{12}).

Some tools are available to manipulate the data in `FltXX_sem2d.dat`:

- 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.

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

- 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` and function `POST/sem2d_read_fault.m` show how to manipulate and plot fault data in Matlab.

4.6.6 Stress glut

For damage and plastic materials, the solver can export the plastic and damage components of the cumulative stress glut tensors defined, respectively, as

$$s_{ij}^p(t) = - \iint 2\mu \epsilon_{ij}^p(t) \, dx \, dz \quad (4.2)$$

$$s_{ij}^d(t) = \iint [\sigma_{ij}(t) - c_{ijkl}^0 \epsilon_{kl}^e(t)] \, dx \, dz \quad (4.3)$$

where ϵ^p is the plastic strain, ϵ^e the elastic strain, σ the absolute stress and c^0 the tensor of elastic moduli of the undamaged medium.

To enable this feature: set `COMPUTE_STRESS_GLUT = .true.` in file `SRC/constants.f90`, then re-compile the code. The stress glut output is exported in the file `stress_glut_sem2d.tab` in 7 columns: time, s_{11}^p , s_{22}^p , s_{12}^p , s_{11}^d , s_{22}^d , s_{12}^d .

4.6.7 Energies

The solver can export the cumulative plastic energy, the kinetic energy and the total change of elastic energy, defined respectively by

$$E^p(t) = \iiint_0^t \sigma_{ij}(t') \dot{\epsilon}_{ij}^p(t') \, dx \, dz \, dt' \quad (4.4)$$

$$E^k(t) = \iint \rho v_i^2(t) \, dx \, dz \quad (4.5)$$

$$E^e(t) = \iint U[\epsilon^e(t)] - U[\epsilon^e(0)] \, dx \, dz \quad (4.6)$$

where U is the elastic potential and summation over subindices is implied.

To enable this feature: set `COMPUTE_ENERGIES = .true.` in file `SRC/constants.f90`, then re-compile the code. The energy output is exported in the file `energy_sem2d.tab` in 4 columns: time, E^p , E^k and E_e .

4.6.8 Matlab utilities

A range of functions and sample scripts for Matlab are available to read, manipulate and plot output data. Add the directory `POST/` to your Matlab path (`addpath`). For an overview of existing utilities, type `help POST`:

SEM2DPACK/POST provides Matlab utilities for the manipulation and visualization of SEM2DPACK simulations results.

Reading simulation data:

SEM2D_READ_SPECGRID reads a spectral element grid
SEM2D_SNAPSHOT_READ reads snapshot data
SEM2D_READ_SEIS reads seismogram data
SEM2D_READ_FAULT reads fault data

Data manipulation:

SEM2D_EXTRACT_POINT extracts field values at an arbitrary point
SEM2D_EXTRACT_LINE extracts field values along a vertical or horizontal line
ARIAS_INTENSITY computes Arias Intensity and Significant Duration
RESPONSE_SPECTRUM computes response spectra (peak dynamic response of single-degree-of-freedom systems)

Data visualization:

SEM2D_PLOT_GRID plots a spectral element grid
SEM2D_SNAPSHOT_PLOT plots snapshot data
SEM2D_SNAPSHOT_GUI interactively plots snapshot data
SEM2D_SNAPSHOT_MOVIE makes an animation of snapshot data
PLOT_MODEL plots velocity and density model
PLOT_SEIS plots multiple seismograms
PLOT_FRONTS space-time plot of rupture front and process zone tail
SAMPLE_FAULT example of visualization of fault data
SAMPLE_SEIS example of visualization of seismogram data

Miscellaneous tools:

XCORRSHIFT cross-correlation time-delay measurement
SPECSHIFT signal time shift by non-integer lag via spectral domain
SPECFILTER zero-phase Butterworth filter via spectral domain

Chapter 5

Adding features to SEM2D (notes for advanced users)

Sometimes you will need to add new capabilities to the SEM2DPACK solver, by modifying the program. The following notes are intended to guide you through this process. We will not give here a comprehensive description of the code architecture, only enough details to get you started in performing safely the most usual and evident modifications.

5.1 Overview of the code architecture

[... in progresss ...]

This code uses a mixture of procedural (imperative) and object-oriented paradigms. Historically, it evolved from a purely procedural code.

Extensive use of modularity.

Object Oriented Programming (OOP) features (principles) applied in this code: encapsulation, classes, static polymorphism. These are not applied everywhere in the code, for different reasons: reusage of legacy code, performance, difficulty related to the limits of Fortran 90, or sections of code yet to be updated.

Added cost of structures containing pointer components: the possibility of pointer aliasing prevents more aggressive compiler optimizations and adds overhead for safety checks.

5.2 Accessible areas of the code

Some areas of the code have been written in such a way that a moderately experienced Fortran 95 programmer, with a limited understanding of the code architecture, can introduce new fea-

tures without breaking the whole system. This is achieved through modularity, encapsulation and templates. The modifications that are currently accessible are:

- boundary conditions, see `bc_gen.f90`
- material rheology, see `mat_gen.f90`
- source time functions, see `stf_gen.f90`
- spatial distributions, see `distribution_general.f90`

The source files listed above contain step-by-step instructions, just follow the comments starting by `!!`.

Chapter 6

Frequently Asked Questions

6.1 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` under bash or `limit stacksize unlimited` under csh. Place this command in your startup files (`.login`, `.bashrc` or `.cshrc`).

Instabilities on very distorted elements

Very distorted elements (with very small or very large angles) are usual close to wedges of sedimentary basins, fault branching points, etc. In general, distorted elements are less stable than square elements: spurious motions with exponentially increasing amplitude might appear in their vicinity. In most cases these instabilities can be suppressed by reducing the **Courant** input parameter. There is currently no simple recipe to determine the maximum value of this parameter, so trial and error is required.

6.2 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.
- Ampuero, J.-P. and Y. Ben-Zion (2008). Cracks, pulses and macroscopic asymmetry of dynamic rupture on a bimaterial interface with velocity-weakening friction. *Geophys. J. Int.* 173(2), 674–692, doi:10.1111/j.1365–246X.2008.03736.x.
- Andrews, D. (1999). Test of two methods for faulting in finite difference calculations. *Bull. Seis. Soc. Am.* 89, 931–937.
- Andrews, D. J. (2005). Rupture dynamics with energy loss outside the slip zone. *J. Geophys. Res.* 110(B1), B01307, doi:10.1029/2004JB003191.
- Ben-Zion, Y. and V. Lyakhovsky (2006). Analysis of aftershocks in a lithospheric model with seismogenic zone governed by damage rheology. *Geophys. J. Int.* 165(1), 197–210.
- Clayton, R. and B. Engquist (1977). Absorbing boundary conditions for acoustic and elastic wave equations. *Bull. Seis. Soc. Am.* 67(6), 1529–1540.
- De la Puente, J., M. Käser, M. Dumbser, and H. Igel (2007). An arbitrary high order discontinuous galerkin method for elastic waves on unstructured meshes iv: Anisotropy. *Geophys. J. Int.* 169(3), 1210–1228, doi:10.1111/j.1365–246X.2007.03381.x.
- Hamiel, Y., Y. F. Liu, V. Lyakhovsky, Y. Ben-Zion, and D. Lockner (2004). A viscoelastic damage model with applications to stable and unstable fracturing. *Geophys. J. Int.* 159(3), 1155–1165.
- Haney, M., R. Snieder, J.-P. Ampuero, and R. Hofmann (2007). Spectral element modelling of fault-plane reflections arising from fluid pressure distributions. *Geophys. J. Int.* 170(2), 933–951.
- Kaneko, Y., N. Lapusta, and J.-P. Ampuero (2008). Spectral element modeling of earthquake rupture on rate-and-state faults: Effects of velocity-strengthening friction at shallow depths. *submitted to J. Geophys. Res.* xx(xx), xxx–xxx.
- 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., C. Barnes, and J. Tromp (2000). Simulation of anisotropic wave propagation based upon a spectral element method. *Geophysics* 65(4), 1251–1260, doi:10.1190/1.1444816.
- 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.
- Lyakhovsky, V., Y. Ben-Zion, and A. Agnon (1997). Distributed damage, faulting, and friction. *J. Geophys. Res.* 102(B12), 27635–27649.
- Lyakhovsky, V., Y. Hamiel, J.-P. Ampuero, and Y. Ben-Zion (2009). Non-linear damage rheology and wave resonance in rocks. *Geophys. J. Int.* 178(2), 910–920, doi: 10.1111/j.1365-246X.2009.04205.x.
- Madariaga, R., J.-P. Ampuero, and M. Adda-Bedia (2006). Seismic radiation from simple models of earthquakes. In A. McGarr, R. Abercrombie, H. Kanamori, and G. di Toro (Eds.), *Earthquakes: Radiated Energy and the Physics of Earthquake Faulting*, Volume 170 of *Geophysical Monograph*, pp. 223–236. Am. Geophys. Union.
- Patera, A. (1984). A spectral element method for fluid dynamics: laminar flow in a channel expansion. *J. Comp. Phys.* 54, 468–488.
- Rubin, A. M. and J. P. Ampuero (2007). Aftershock asymmetry on a bimaterial interface. *J. Geophys. Res.* 112, B05307.
- Stacey, R. (1988). Improved transparent boundary formulations for the elastic-wave equation. *Bull. Seism. Soc. Am.* 78(6), 2089–2097.
- 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.