SEM Documentation

Version 2013.02.0

L. Aubry

03 February 2014

Contents

CHAPTER 1

Introduction

SEM est un code de propagation d'onde sismique, en 2D et 3D, fondé sur la méthode des éléments spectraux ([PRI94], [FAC97], [KOM98], [SER98], [KOM99]).

Cette version prend en compte la propagation dans des milieux hétérogènes, à géométrie complexe (topographie de surface, interfaces).

Un outil de prétraitement, permet de convertir et partitionner (avec **Metis**) des maillages au format **Abaqus** ou **Ideas** (UNV).

Des sources ponctuelles ou planaires peuvent être introduites. Les conditions d'absorption en bord de domaine utilisent la méthode *Perfectly Matched Layer (PML)* décrite dans [FES05].

Le code est écrit majoritairement en Fortran 90. Il utilise les librairies BLAS, et MPI.

1.1 Notations

Exemple des notations utilisées dans ce document :

- Un programme ou logiciel : **CMake**
- Une commande Unix (`cmake`), un nom de fichier (`/usr/bin/python`) ou un texte à saisir ("saisissez le mot-clef `false`").
- Un terme en anglais : *remote control*
- Une variable d'environnement : `HDF5_ROOT`

Parfois le programme et sa commande Unix ont le même nom, on essaira de faire la distinction, par exemple :

"Pour configurer un programme utilisant **CMake** il faut taper la commande `cmake`."

CHAPTER 2

NEWS

2.1 Version 2013.12

- Stabilisation des éléments fluides et couplage fluide/solide
- Nouvelles fonctions d'évolution temporelle des sources (square, tanh, ...)
- La limite maximum du nombre de processeurs gérés par le mailleur est portée à 8192 (1024 précédement). La consommation mémoire excessive en (nombre de processeurs * nombre de mailles) est résolue. On a maintenant une consommation proportionnelle à (nombre de mailles x nombre de processeurs voisins).
- un paramètre amplitude global pour toutes les fonctions temporelles est ajouté.
- la dépendance sur HDF5 est maintenant obligatoire.
- Le mailleur accepte un format hdf5 (semblable àau format UNV) en entrée. Permet de gérer de gros maillages beaucoup trop long à lire dans un format texte.
- Limitation du nombre de sorties texte du code pour passer des codes sur un grand nombre de processeurs.
- bugfix: le flag mpml n'était plus pris en compte.
- bugfix: le mailleur ne libérait pas immédiatement les ressources HDF5, ce qui induisait des temps très long de flush en fin de job. Ce temps étant compté dans l'épilogue MPI, le processus était tué avant la fin.
- bugfix: les paramètres d'attenuation n'étaient pas correctement sauvegardé lors des protection reprise.
- Les sorties ont été mutualisées par groupe de processeurs, permettant d'avoir une sortie par noeud de calcul.

, plutot qu'une sortie par processus MPI.

- bugfix : pour les fluides, les excitations sont pondérées par lambda et plus rho
- fluide : on assure la continuité de rho*phi et non plus phi.
- mailleur: on peut mailler un milieu stratifié simple

- SEM2D : mutualisation de code, utilisation du nouveau format de fichier d'entrée commun avec SEM3D.
- Nouveaux champs en sortie des snapshots : pression et accélération
- optimisation du calcul des forces solides sans Acoef. Le calcul des dérivées spatiales a maintenant des cas particuliers pour ngll=5 et 7. Au delà de 10, l'ancienne méthode avec DGEMM optimisée (MKL) devient plus intéressante.
- correction fuite mémoire (initiale, stable) dans l'allocation des capteurs.

2.2 Version 2013.04

Cette version résulte de l'intégration dans **RegSEM.U** de :

- des modifications apportées par la version interne CEA,
- des éléments fluides développés dans une autre version issue de **RegSEM.U**,
- de nouveaux développements destinés à simplifier l'utilisation et la maintenance du code.

2.2.1 Les nouveautés

On liste ici les nouvelles fonctionnalités par rapport à **RegSEM.U**.

Fonctionnalités du code :

- (**SEM3D**) Introduction d'éléments de type fluide, avec couplage fluide solide.
- Introduction d'un mécanisme d'amortissement sismique. On spécifie Qp et Qs dans le fichier matériau. La bande de fréquence et le paramétrage du filtre est déterminé par le fichier de configuration.
- Nouvelles formes d'onde pour les sources (Benchmark E2VP, Benchmark SPICE, sinus).
- Une variante des PML (MPML) avec son paramètre associé a été introduite. Ceci afin de régler des problèmes d'instabilités constatés sur certains cas.
- Un mode couplage optionnel avec un code externe (pour l'instant Mka3D).
- On peut maintenant faire des sorties snapshots partielles. Le fichier `input.spec` permet de décrire simplement une sélection de mailles à inclure dans les sorties.

Entrées/sorties :

- (MESH) Lecture des maillages au format unv.
- (**SEM3D**, **SEM2D**) Un nouveau format de fichier d'entrée (input.spec) :

  L'ancien format était très confu : une liste de valeurs lues de manière aveugle par les codes. Chaque code lisait ses paramètres dans un ordre pré-établi. Il était impossible de réutiliser un fichier de config d'une version à l'autre.

  Désormais les paramètres sont identifiés par des mots-clefs. Ainsi un paramètre inconnu est soit ignoré soit génère une erreur.

  Les sources sont décrites dans ce format.

- Les snapshots sont au format **HDF5** :

  Le code génère en plus des fichiers **HDF5**, un fichier XML (format XDMF) qui permet d'ouvrir directement les sorties dans **Paraview** ou **Ensight** (v10).

- Les maillages en entrée sont également au format **HDF5** :

  Des problèmes de numérotation apparaissaient avec des gros maillages (utilisation du format `I6` pour les entiers). De plus, chacune des versions utilisait une variante subtile du même format texte (une ligne d'espacement pour l'un, un champ supplémentaire pour une autre...).

  Les identifiants sont maintenant des entiers 32 bits permettant de décrire 2 milliards de noeuds uniques, et le format utilise par défaut la compression gzip.

- Nouveau format pour le fichier des capteurs/traces :

  On a conservé le format de la version CEA, plus général. Dans une prochaine version ce fichier migrera vers un format semblable à celui de `input.spec`.

- Le format des backups est désormais **HDF5** (protection/reprise).

  Ce développement à été effectué pour faire passer un cas HPC. Le temps de création d'un backup pour ce cas est passé de 2H à 5min.

Optimisations :

- Optimisation des communications :

  L'algorithme d'échange inter-processeur a été entièrement revu pour utiliser des communications asynchrones. Il n'y a plus de risque d'interblocage occasionnel et les performances sont accrues.

- Optimisation de la consomation mémoire :

  Les mailles non-PML consommaient inutilement de la mémoire en stockant des pointeurs (non-alloués) vers des tableaux concernant uniquement les mailles PML.

  Une structure spécifique PML a été introduite. Celle-ci n'est allouée qu'au besoin uniquement pour les éléments contenant des PML. La mémoire utilisée est réduite à l'espace d'un seul pointeur par élément au lieu d'une dizaine.

- L'utilisation de la librairie **HDF5** permet d'optimiser grandement les Entrées/Sorties pour les gros cas de calcul.

Autres :

- Améliorations du mailleur intégré :

  On utilise **Metis** 5.x comme partitionneur. Ceci permet d'utiliser une topologie connectant toutes les mailles adjacentes (ayant au moins un vertex commun) contrairement à la version précédente qui ne considérait que les faces.

  Le mailleur génère ses maillages au format **HDF5** attendu par SEM.

  De nombreuses optimisations et restructurations du code ont été effectuées accélérant le traitement.

- Introduction d'un répertoire de cas tests de non-régression et de benchmarks.

  Les tests **SEM3D** se trouvent dans `SEM3D/TESTS`.

- Compilation des sources avec **CMake** :

CMake est un outil (comme autotools) permettant de générer des Makefiles. (voir *Installation du code SEM3D*).

- Correction des FPML.
- (**SEM3D**) : le code a été factorisé (suppression des duplications, réorganisations, simplifications) en plusieurs endroits.

2.2.2 Evolutions futures

Certaines fonctionnalités sont prévues (voire déjà disponibles dans le code) mais n'ont pas encore été finalisées, intégrées ou correctement testées :

- Description de gradient de propriétés dans les matériaux. Le code de la version CEA a été intégré, mais la description des matériaux dans le fichier de configuration n'a pas encore été effectuée.

  La nouvelle description des gradients et le nouveau format du fichier matériaux seront développés dans une future version.

- Description des conditions de Neumann. Le code existe, il n'a pas été testé. Il sera intégré dans le fichier de configuration au nouveau format dans une prochaine version.
- Description des capteurs : la prochaine version utilisera une syntaxe semblable à celle du fichier `input.spec` pour la description des capteurs.
- Anisotropie : le code pour gérer des matériaux anisotropes existe, mais il n'y a rien dans la syntaxe actuelle du fichier de description des matériaux qui permette de définir un milieu anisotrope. Là encore, cela sera intégré dans la prochaine version lors de la refonte du fichier de description des matériaux.

2.2.3 Notes importantes

Le code source est versionné avec **Git** et livré dans une archive contenant :

- SEM version 3D
- SEM version 2D
- MESH : un outil de préparation de maillages 3D pour **SEM3D** (l'équivalent 2D sera intégré dans une prochaine version).
- La librairie **HDF5** est devenue une dépendance obligatoire (www.hdfgroup.org).

  Cette librairie permet le stockage efficace de gros volume de données. Son utilisation permet le posttraitement immédiat des snapshot avec Paraview ou Ensight. Les données produites sont également lisibles facilement avec Matlab et Python.

- Le schéma en temps a été simplifié (Les paramètres beta/gamma de l'algorithme de Newmark ne sont plus modifiables).

  Ils pourront être réintroduits une fois réglé le problème de synchronisation avec les forces de couplage externes.

- Bien que les deux méthodes continuent de coéxister, le calcul des forces utilisant le tableau `Acoeff` a été désactivé dans cette version. Le code est plus lisible mais moins rapide.

  On étudiera comment obtenir le meilleur des deux méthodes dans une prochaine version.

CHAPTER 3

Prise en main de SEM3D

3.1 Introduction

Ce chapitre présente la chaine de calcul SEM, insistant plus particulièrement sur SEM3D.

La chaîne logicielle SEM contient le code de simulation (`sem2d.exe` et `sem3d.exe`, ainsi qu'un outil de partitionnement et préparation de maillage `mesher`).

L'outil `mesher` permet de partitionner des maillages complexes au format abaqus, UNV ou encore un format spécifique simple (équivalent du format UNV dans un fichier HDF5).

Outre la paramétrisation du code SEM3D, la plus grosse difficulté dans l'utilisation du code concerne la création des maillages, qui doit être effectuée par des outils spécialisés comme Cubit. Pour des maillages simples : stratifiés, avec topographie, on peut utiliser la suite d'outils *meshtools*.

3.1.1 Les équations du mouvement

SEM résout la propagation d'onde élastique dans un milieu décrit par l'équation locale du mouvement reliant le déplacement u en chaque point matériel, les contraintes σ et les forces extérieures $\vec{f}$:

$$\rho \frac{\partial^2 u}{\partial t^2} = \nabla . \sigma + \vec{f}$$

Avec en élasticité linéaire : $\sigma = C : \nabla u$, où C est le tenseur élastique d'ordre 4.

Pour l'instant les milieux de propagations décrits dans SEM sont considérés isotropes. Le code est prévu pour gérer les milieux anisotropes, mais il n'existe pas de manière simple de gérer la mise en données.

3.1.2 Formulation éléments finis

SEM est un code éléments finis, basé sur une formulation spectrale (d'où son nom). Le champ de déplacement u est décrit dans chaque élément, ou maille, sur une base de polynômes de Lagrange d'ordre N (N défini comme paramètre).

Pour obtenir une convergence spectrale, ces polynômes de Lagrange sont définis sur les points de Gauss-Lobato-Legendre (GLL) de chaque éléments (voir *Position des points de Gauss-Lobato-Legendre sur un élément 2D d'ordre 9*).

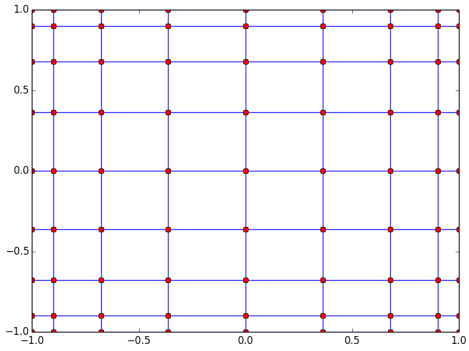

Figure 3.1: Position des points de Gauss-Lobato-Legendre sur un élément 2D d'ordre 9

Les éléments traités sont des quadrangles en 2D et des Hexaèdres en 3D. Si $\Phi_i(x)$ est le polynôme de Lagrange valant 1 au point GLL x_i, le champ de déplacement $u(x, y, z)$ de l'élément s'exprime sur la base tensorisée $\Phi \otimes \Phi \otimes \Phi$:

$$u(x, y, z) = \sum_{i,j,k} U_{i,j,k}.\Phi_i(x).\Phi_j(y).\Phi_k(z)$$

Ainsi, sur un élément d'ordre 5, la composante *x* du champ de déplacement, est décrite par un vecteur de 125 éléments $U_{i,j,k}$.

La figure *Polynômes de Lagrange d'ordre 9.* montre la forme des polynôme de Lagrange d'ordre 9, la base tensorisée de dimension 2D est représentée *Quelques fonctions de forme d'un élément 2D d'ordre 9*

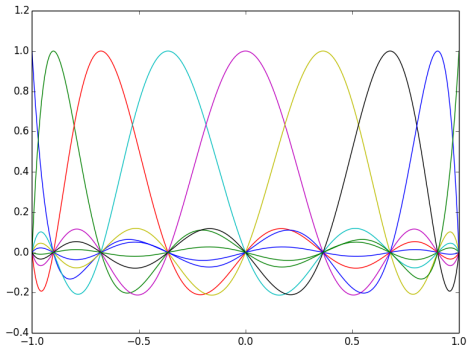

Figure 3.2: Polynômes de Lagrange d'ordre 9.

Le champ de déplacement est continu entre deux éléments adjacents, et le maillage géré par SEM doit être conforme (toutes mailles se touchant ont en commun un sommet, ou une arête ou une face complète). De plus l'ordre en X, Y ou Z de chaque maille doit assurer la conformité au niveau des points GLL en commun.

Enfin, dans ce qui précède, on a présenté la formulation de manière simplifiée, sur des mailles cubiques, alignées en *x,y,z*. En pratique SEM peut gérer un maillage hexaédrique quelconque (mais conforme), composé de mailles parallélépipédiques. Dans chaque élément le code se ramène à une base locale $\phi_i(x).\phi_j(y).\phi_k(z)$ par changement de variable de la fonction de base Φ depuis la maille vers un élément de référence sur le segment [-1.,1.] .

Enfin une des originalités de la méthode, provient du choix de quadrature pour l'évaluation numérique des intégrales apparaissant dans la formulation élément finis.

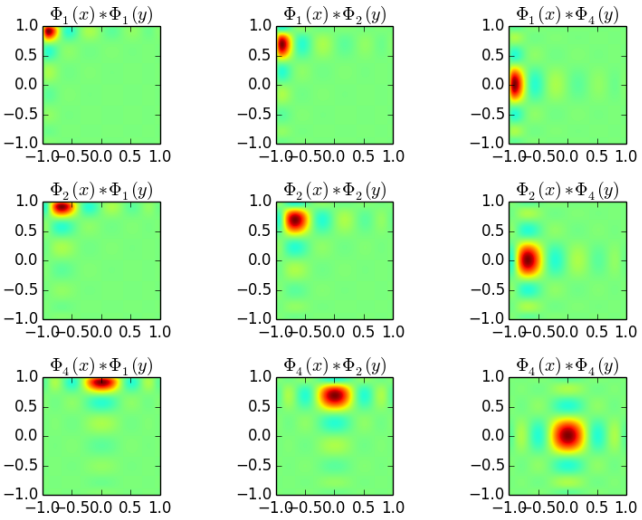

Figure 3.3: Quelques fonctions de forme d'un élément 2D d'ordre 9

On passe d'abord à la formulation faible en multipliant l'équation locale par une fonction quelconque *w* et en intégrant (produit scalaire dans $\mathcal{L}^2$)

$$\int w.\rho\frac{\partial^2 u}{\partial t^2}\vec{dx} = \int w.(\nabla.(C:\nabla u) + \vec{f}).\vec{dx}$$

En exprimant *w* et *u* sur la même base discrète $\Phi_i(x,y,z)$ (ici *i* indexe **toutes** les fonctions de base de tous les éléments).

$$\sum_{i,j} w_i.\rho\frac{\partial^2 u_j}{\partial t^2}\int \Phi_i\Phi_j\vec{dx} = \sum w_i.u_j.(\nabla.(C:\nabla\Phi_j) + f_j\Phi_j).\vec{dx}$$

Cette dernière équation apparaît alors sous la forme classique de l'approximation de Galerkin : $a(u,w) = f(w)$ avec a une forme bilinéaire.

Sans aller jusqu'au bout des développements, on voit qu'il apparaît une matrice $M_{i,j} = \int \Phi_i\Phi_j\vec{dx}$, que l'on doit inverser si on veut obtenir une expression de $\frac{\partial^2 u_j}{\partial t^2}$.

Les produits scalaires entre fonctions Φ_i qui ne partagent pas le même élément support sont nuls par construction. Mais au sein d'un éléments, les polynômes de Lagrange ne sont pas orthogonaux. La méthode SEM utilise astucieusement une quadrature basée sur les mêmes points de Gauss que les noeuds de définitions des fonctions de base. Cela introduit bien sûr une approximation de l'intégrale, mais le résultat est que le produit scalaire discret utilisé rend orthogonale les fonctions Φ_i ayant le même élément support.

3.1.3 Conditions de bord

La condition naturelle d'un bord en élément fini est d'être une surface libre, donc réfléchissante pour les ondes. Pour simuler des milieux ouverts, SEM implémente un type d'élément dit PML (Perfectly Matched Layer) pour simuler un milieu ouvert infini en bordure d'un domaine.

3.1.4 Intégration temporelle

Le schéma d'intégration est un schéma de Newmark explicite.

Le pas de temps d'intégration dans SEM est calculé automatiquement à partir du nombre de Courant $\mathcal{C} < 1$ (paramètre de configuration) selon :

$$\Delta t = \mathcal{C} \frac{\min \Delta x}{\max Velocity}$$

Attention:

> Des mailles trop petites, ou des vitesses de propagation trop importantes vont faire chuter le pas de temps.

3.1.5 Résolution spatiale

Le maillage doit également être suffisement résolu pour capturer les fréquences spatiales du signal que l'on veut propager. On considère que 10 points GLL par longueur d'onde sont suffisant.

Augmenter l'ordre des éléments est donc un moyen d'obtenir une résolution spatiale correcte avec un maillage donné. La convergence spatiale étant rapide, augmenter l'ordre devrait permettre de baisser le nombre de points par longueur d'onde nécessaire, mais cela augmente doublement les coûts de calcul :

- la complexité est en N^3 par points GLL,
- le pas de temps est proportionnel à $\frac{1}{\min \Delta x}$, le pas d'espace $\min \Delta x$ diminuant avec l'ordre des éléments (On voit sur *Position des points de Gauss-Lobato-Legendre sur un élément 2D d'ordre 9* comment les points de Gauss se ressèrent vers les bords avec l'augmentation de l'ordre.

3.1.6 Atténuation

Un mécanisme d'atténuation sismique des ondes P et S est implémenté, sous forme d'une série de filtres répartis sur une bande de fréquence. (voir [KOM98])

3.1.7 Description des sorties

Les résultats de simulation peuvent être obtenus sous deux formes :

- Des instantanés (*snapshot*) des champs obtenus sur tous les points GLL, ou sur un sous-partie, à une fréquence données. Ces sorties sont en général assez lourdes et ne peuvent être trop fréquentes.
- Des sorties *capteurs*, pour un ou plusieurs points du maillage, on sort les valeurs du champ toutes les N itérations de calcul.

Les champs disponibles sont :

Champ	Milieu	Snapshot	Capteurs
Déplacement	S	Oui	Oui
Vitesse	S/F	Oui	Oui
Accélération	S/F	Oui	Non
Pression	S/F	Oui	Non

Pour les instantanés, il existe un mécanisme de sélection de mailles qui permet de ne sauvegarder qu'une partie du maillage. Cependant on ne peut sélectionner que des mailles complètes (donc avec tous ses points GLL), et pour l'instant, on ne peut pas, sauf en post-traitement, réinterpoler les fonctions de formes sur un maillage plus grossier.

3.2 Présentation des outils

Deux exécutables sont impliqués directement dans l'utilisation de SEM :

- `mesher` et `sem3d.exe` pour le cas 3D,
- `sem2d.exe` pour le cas 2D, il n'existe pas encore d'outil de partitionnement simple à utiliser.

`mesher` transforme un maillage d'entrée en un maillage partitionné utilisable par SEM. On peut lui fournir différents formats :

- Un maillage au format *Abacus* (d'extension `.aba`)
- Un maillage au format *UNV*, (aussi connu sous le nom *IDEAS*) d'extension `.unv`, contenant des hexaèdre pour la 3D.
- Un maillage au format *HDF5*, spécifique, dont la structure est décrite en détail dans Format HDF5, contenant 3 tables :
  - `Elements` : un tableau de NE x 8 entiers de 0 à (NN-1) faisant référence aux noeuds.
  - `Nodes` ; un tableau de NN x 3 de réels, les coordonnées des noeuds
  - `Mat` : un tableau de NE entiers, contenant le numéro matériau à associer à chaque maille.
- Le quatrième format est simplement la description d'un maillage cartésien, pour lequel on entre manuellement les coordonnées et la subdivision de la grille souhaitée.

L'outil mailleur, en plus de ses entrées en ligne de commande, s'appuie sur un fichier externe `mat.dat`, donnant quelques informations sur le maillage à générer : nombre de matériaus, présence d'éléments PML, type de matériau (solide ou fluide).

3.2.1 Préparation d'un cas de calcul

Pour lancer un calcul SEM, il faut se placer dans le répertoire du cas et y placer les fichiers nécéssaires à son exécution. L'arborescence doit être la suivante

```
CAS/
|- input.spec
|- material.input
|- sem/
|  |- mesh4spec.0000
|  |- ...
|  |- mesh4spec.NNNN
|- capteurs.dat
```

input.spec:

> Ce fichier contient la configuration du code : - paramètres d'intégration temporelle, temps physique du calcul, - description de la ou des sources, - description des sorties capteurs, - description des sorties snapshots.

material.input:

> Ce fichier contient la description de chaque matériau : ρ, V_p, V_s, un nombre de points GLL par direction de la maille de référence.

capteurs.dat

> Contient une description des sorties capteurs souhaitées.

Le fichier `input.spec` est décrit en détail dans la section Description des paramètres de SEM3D.

Des exemples de fichiers *material.input* et *capteurs.dat* sont disponibles dans les tests du code. Ces derniers sont de simples tables de paramètres.

3.3 Exemples de modélisation avec SEM3D

3.3.1 Maillage uniforme avec PML

On commence par un premier exemple de grille cartésienne avec une source ponctuelle.

Le fichier `mat.dat` doit contenir (les commentaires, après le # sont facultatifs)

```
1  # number of non PML materials
F  # Milieu stratifié F: non T: oui
1  # PMLs? 0: no, 1: yes
1 1  # PMLs on top? at the bottom? (0: no, 1: yes)
S
```

On lance l'exécutable `mesher`, et on lui indique les informations suivantes :

- Nombre de procésseurs : 4
- Construction du modèle matériaux et maillage : 1 (Oui)
- Choix d'une grille : 1 (On the fly : sur la mouche)
- Saisie des coordonnées et taille de maille :
  - X : -100, 500
  - Y : -100, 500
  - Z : -100, 500
  - DX, DY, DZ : 50
- Choix de 8 noeuds par maille : 1 (Les mailles quadratiques à 27 noeuds sont en développement)

L'outil va alors générer 4 fichiers nommés `mesh4spec.000N.h5` (N=0,1,2,3) contenant les maillages et informations de communication des 4 partitions.

3.3.2 Lancement du cas

Il faut d'abord préparer le répertoire du CAS : y copier les fichiers `input.spec`, `material.input`, `capteurs.dat`, et placer les fichiers `mesh4spec.NNNN` dans le sous-répertoire `sem/`.

CHAPTER 4

Description des paramètres de SEM3D

4.1 Format

Le format de fichier SEM3D a été changé pour plus de souplesse et pour éviter des erreurs de saisie.

Le nouveau format de fichier est de la forme suivante (exemple)

```
mot_clef = valeur;  # commentaire
# commentaire
section {
   mot_clef2 = "chaine";
};
mot_clef3 = v1 v2 v3; # un vecteur de valeurs
```

Les valeurs sont des entiers, des flottants, des booléens, des chaines ou des mot-clefs.

Une chaine est une suite de caractères entre guillemets ("), un mot clef est une suite de caractères alphanumeriques commençant par une lettre, et comportant des lettres, des chiffres ou le caractère souligné (_).

Les sections peuvent apparaître plusieurs fois (par exemple la section `source`).

Les paramètres peuvent apparaître dans un ordre quelconque au sein d'une section (ou du corps principal). Un paramètre valide peut-être ignoré si il n'est pas activé par un autre paramètre : par exemple on peut désactiver les snapshots, tout en laissant le paramètre nombre d'itération entre snapshot.

Les mots-clef pouvant être utilisés dans le fichier (niveau 0, hors toute section) sont décrits ici :

Mot-clef	Type	Valeur par défaut	Description
amortissement	section	n/a	Description de l'amortissement
mat_file	chaîne	"material.input"	Nom du fichier de description des matériaux
mesh_file	chaîne	"mesh4spec"	Nom de base des fichiers maillage
mpml_atn_param	réel	0.0	Coéfficient d'amortissement MPML (et activation MPML si non nul)
prorep	bool	false	Reprise d'un calcul précédent
prorep_iter	entier	n/a	Numéro de la protection pour reprendre le calcul
run_name	chaîne	""	Titre de la simulation
snapshots	section	n/a	Description des paramètres de sauvegarde des snapshots
save_traces	bool	false	Activation des capteurs
traces_format	kw	text	Format des sorties capteurs `text` ou `hdf5`
sim_time	réel	aucune	Durée (temps physique) de la simulation
source	section	n/a	Description d'une source (peut apparaître plusieurs fois)
station_file	chaîne	"capteurs.dat"	Fichier de description des capteurs
time_scheme	section	n/a	Section de description du schéma d'intégration en temps
verbose_level	entier		

Les paramètres suivants sont reconnus mais non utilisés dans cette version :

Mot-clef	Type	Valeur par défaut	Description
anisotropy	bool	n/a	Description de l'anisotropie
gradient	section	n/a	Description des gradients
model	kw	–	CUB|homo|prem|3D_berkeley
neumann	bool		.
traces_interval	entier		.
traces_format	kw		

Description de la section `amortissement` :

Mot-clef	Type	Valeur par défaut	Description
nsolids	entier	0	Nombre de mécanismes. 0 signifie désactivation.
atn_band	réel(2)	n/a	Période max et min à atténuer
atn_period	réel	n/a	Période centrale (un mécanisme aura cette valeur centrale)

Description de la section `time_scheme` :

Mot-clef	Type	Valeur par défaut	Description
accel_scheme	bool		Schéma en temps
veloc_scheme	bool		Schéma en vitesse
alpha	réel		Paramètre α d'intégration de Newmark
beta	réel		Paramètre β d'intégration de Newmark
gamma	réel		Paramètre γ d'intégration de Newmark
courant	réel	0.2	Nombre de courant. Le calcul du pas de temps en dépend.

Description de la section `source` :

Mot-clef	Type	Valeur par défaut	Description
coords	réel(3)	0 0 0	Position de la source
type	kw	–	Type spatial: impulse|moment|fluidpulse
dir	kw	–	Direction pour le type impulse ou fluidpulse (val: x|y|z)
func	kw	–	Type temporel (voir Les sources ci-dessous)
moment	réel(6)	–	Moment xx yy zz xy yx xz pour le type moment
tau	réel	–	Un temps caractéristique τ
freq	réel	–	Une fréquence f_c
band	réel(4)	–	Description des bornes f_1, f_2, f_3, f_4 pour tf_heaviside
ts	réel	–	Un offset de temps t_0
gamma	réel	–	
time_file	chaîne	–	Fichier contenant la source
amplitude	réel	–	Facteur multiplicatif appliqué à la source temporelle

Description de la section `snapshots` :

Mot-clef	Type	Valeur par défaut	Description
save_snap	bool	false	Sauvegarde des snapshots
save_interval	réel	–	Interval (temps physique) de sauvegarde des snapshots
select	voir note	–	Sélection des éléments à inclure dans les snapshots
deselect	voir note	–	Désélection des éléments à inclure dans les snapshots
group_outputs	entier	32	Écriture d'un fichier sortie par *group_outputs* processeurs

Note: Par défaut, les snapshots incluent toutes les mailles. Le format de la commande select/deselect est décrit ci-dessous.

On peut choisir de sélectionner ou déselectionner des mailles pour les inclure ou les exclure des sorties.

Il y a pour l'instant deux critères de sélection : le numéro du matériau ou la localisation absolue.

Les commandes de sélection/déselection sont appliquées dans l'ordre du fichier `input.spec`.

La syntaxe de la commande est :

```
[de]select (all|material = NN|box = x0 y0 z0 x1 y1 z1) ;
```

Ainsi :

```
deselect all;
select material = 1;
selec box = -500 -10 -10 500 10 10;
```

Va déselectionner tous les éléments, puis resélectionner tous les éléments ayant le matériau 1, ainsi que tous les éléments dont le centre se situe dans la boite spécifiée.

Autre exemple :

```
select all;  # Inutile car par défaut
deselect material  = 5;
deselect material  = 6;
deselect material  = 7;
```

Cette description va simplement exclure les matériaux 5, 6 et 7 des sorties.

4.2 Exemple

Le fichier suivant correspond à celui d'un cas test :

```
# -*- mode: perl -*-
run_name = "Run_3D_trial";

# duration of the run
sim_time = 1.0;
mesh_file = "mesh4spec"; # input mesh file
mat_file = "material.input";

snapshots {
  save_snap = true;
  snap_interval = 0.01;
  deselect all;
  select material = 1;
  select box = -10 -10 -10 10 10 10;
};
save_traces = true;
# Fichier de description des capteurs
station_file = "file_station";


source {                  # introduce a source
  # coordinates of the sources
  coords = 0. 0. 0.;
  type = impulse; # Type (1 Impulse, 2 Moment Tensor, fluidpulse)
  dir = x;                        # Direction x,y ou z (only for Impulse)
  func = ricker;           # Function gaussian,ricker,tf_heaviside,gabor,file
  tau = 0.2;               # tau
  freq = 5.;            # source main frequency (only for Ricker)
};


#gradient_file="gradients.dat"  # fichier gradient

time_scheme {
  accel_scheme = false;  # Acceleration scheme for Newmark
  veloc_scheme = true;   # Velocity scheme for Newmark
  alpha = 0.5;           # alpha (Newmark parameter)
  beta = -0.5;           # beta (Newmark parameter)
  gamma = 1;             # gamma (Newmark parameter)
};
```

4.3 Les sources

Les formes d'ondes temporelles des sources sont décrites ci-dessous. Les paramètres sont décrits dans la section `source`. Certains sont calculés :

- f_c : paramètre `freq`
- $T_c = \frac{1}{f_c}$
- τ : paramètre `tau`

- t_0 : paramètre `ts`
- f_1, f_2, f_3, f_4 : décrits par le paramètre (4 composantes) `band`
- γ : paramètre `gamma`

Les fonctions temporelles sont:

- `gaussian` :

$$f(t) = -2(t - t_0) \exp\left(-\frac{(t - t_0)^2}{\tau^2}\right)$$

- `ricker` :

$$f(t) = \left(1 - 2\left(\pi\frac{t - \tau}{T_c}\right)^2\right) \exp\left(-\left(\pi\frac{t - \tau}{T_c}\right)^2\right)$$

- `tf_heaviside` :

$$\begin{aligned}
f(t) &= \mathcal{TF}^{-1}(\phi(\omega)) && (4.1)\\
\phi(\omega) &= \exp(-i\omega\tau).\chi_{f_1,f_2,f_3,f_4}(\frac{\omega}{2\pi}) && (4.2)\\
\chi(f) &= 1 \text{ if } f_2 < f < f_3 && (4.3)\\
&\quad 0 \text{ if } f < f_1 \text{ or } f > f_4 && (4.4)\\
&\quad \frac{1}{2}\left(1 + \cos\left(\pi\frac{f - f_3}{f_4 - f_3}\right)\right) \text{ if } f_3 < f < f_4 && (4.5)\\
&\quad \frac{1}{2}\left(1 + \cos\left(\pi\frac{f - f_2}{f_2 - f_1}\right)\right) \text{ if } f_1 < f < f_2 && (4.6)
\end{aligned}$$

- `gabor` :

$$\sigma(t) = 2\pi f_c(t - t_0)$$
$$f(t) = \exp(-\left(\frac{\sigma(t)}{\gamma}\right)^2) \cos(\sigma(t) + \omega)\tau$$

- `file` : Les données sont lues dans un fichier indiqué par le paramètre `time_file`
- `spice_bench` :

$$f(t) = 1 - (1 + \frac{t}{T_c}) \exp(-\frac{t}{T_c})$$

- `sinus` :

$$f(t) = \sin(2\pi f_c(t - t_0))$$

- `square` : Un carré *arrondi*

$$f(t) = \frac{\exp(2. * \gamma * (x - t_0)) - 1.}{\exp(2. * \gamma * (x - t_0)) + 1} + \frac{\exp(2. * \gamma * (t_0 + \tau - x)) - 1.}{\exp(2. * \gamma * (t_0 + \tau - x)) + 1}$$

CHAPTER 5

Installation du code SEM3D

5.1 Prérequis

Le code **SEM3D** nécessite deux (ou trois) outils externes pour sa compilation :

- **CMake** : www.cmake.org est un générateur de Makefile (comme autoconf/autotools).
- **HDF5** : www.hdfgroup.org est une librairie permettant la gestion de fichier de données de grande taille.

  Si **HDF5** est installé dans un chemin non-standard, utilisez la variable d'environnement `HDF5_ROOT` pour indiquer aux scripts **CMake** de **SEM** où elle se trouve. `HDF5_ROOT` doit pointer sur le préfixe d'installation, c'est à dire le chemin contenant `bin/`, `include/`, etc...

  Les scripts de configurations **CMake** de **SEM** utilise la commande `h5cc -show` pour détecter le paramètrage de la librairie **HDF5**.
- une librairie **MPI** (**OpenMPI** recommandée, mais **Mpich** ou **Intel MPI** doivent être compatibles).

5.2 Compilation

La préparation de la compilation s'effectue avec **CMake** (commandes: `ccmake` ou `cmake`), ensuite la compilation elle-même s'effectue avec la commande `make`.

5.2.1 Organisation des répertoires

Nous recommandons l'organisation suivante (les noms des répertoires sont à titre indicatif) :

- `sem_src` : répertoire contenant les sources,
- `sem_build` : répertoire contenant les binaires (en dehors du répertoire source),
- `sem_debug` : (facultatif) un second répertoire pour une compilation en mode debuggage de SEM.

5.2.2 Préparation de la compilation

La préparation se fait à l'aide de la commande suivante

```
$ cd sem_build
$ ccmake ../sem_src
```

`ccmake` est une commande interactive de **CMake** permettant de paramétrer la compilation. Le paramétrage s'effectue en deux étapes :

- La première étape (configuration) **CMake** va rechercher les librairies nécessaires (HDF5 et OpenMPI)

  Lors de cette étape on peut changer le contenu des variables affichées (chemin des librairies, nom du compilateur, options de compilation).

  Important : C'est à cet endroit qu'il faut préciser le mode de compilation par la variable : `CMAKE_BUILD_TYPE`. On peut saisir : `DEBUG`, `RELEASE`, ou `RELWITHDEBINFO`. Si on ne saisit rien, **SEM** sera compilé avec les options par défaut du compilateur (sans optimisation et sans debuggage avec **gcc**, optimisé sans debuggage avec **ifort**).

  Lorsqu'on change des variables, il faut reconfigurer (touche `c`).

- La seconde étape, la génération des fichiers `Makefile` ne peut se faire que si l'option `g` (*generate and exit*) apparait dans l'interface. Cette option n'apparait que si la dernière étape de configuration n'a pas modifié de variables.

  En effet, il se peut qu'une reconfiguration change d'autres variables (lorsqu'on change le compilateur par exemple), il faut alors lancer la configuration une seconde fois.

  Lorsque l'étape de configuration ne modifie aucune variable, on peut générer les Makefile (touche `g`).

5.2.3 Compilation

Une fois la génération terminée, la compilation se fait simplement par la commande `make`.

Quelques variantes :

- `make help` : affiche toutes les cibles possibles.
- `make -j N` : compile en parallèle avec N processus (on peut en général utiliser N=nombre de processeurs + 1 ou 2).
- `make -j N -k` : compile le plus possible sur N processeur. Ne s'arrête pas à la première erreur de compilation.
- `make VERBOSE=1` : affiche les lignes de commandes exécutées lors de la compilation.

La compilation produit plusieurs exécutables :

- `build_src/SEM2D/sem2d.exe` : Code SEM2D.
- `build_src/SEM3D/sem3d.exe` : Code **SEM3D**.
- `build_src/MESH/mesher` : Outil pour le partitionnement des maillages et la génération au format **HDF5**.

5.2.4 Exécution

Exécution de **SEM3D** monoprocesseur :

```
$ cd rep_du_cas
$ ${chemin_build}/SEM3D/sem3d.exe
```

Exécution de **SEM3D** en MPI :

```
$ cd rep_du_cas
$ mpirun -n 4 ${chemin_build}/SEM3D/sem3d.exe
```

Lancement du générateur de maillage :

```
$ cd rep_du_cas
$ ${chemin_build}/MESH/mesher
```

Ou en mode automatique avec les saisies clavier enregistrées dans le fichier `mesh.input` (c'est le cas des cas tests présent avec les sources de SEM) :

```
$ cd rep_du_cas
$ ${chemin_build}/MESH/mesher < mesh.input
```

CHAPTER 6

Utilisation du gestionnaire de configuration git

6.1 Introduction

- **git** est disponible sur Unix (Linux, MacOS) (gitscm.org) et Windows (code.google.com/p/tortoisegit/)

Les commandes de ce tutoriel concerne la version unix, mais sont également valables si vous utilisez le *shell* **git** depuis Windows.

- La commande `gitk` ouvre une fenêtre permettant de visualiser les derniers *commits*, la branche courante, les fichiers modifiés, ... On l'exécute en faisant

```
gitk --all &
```

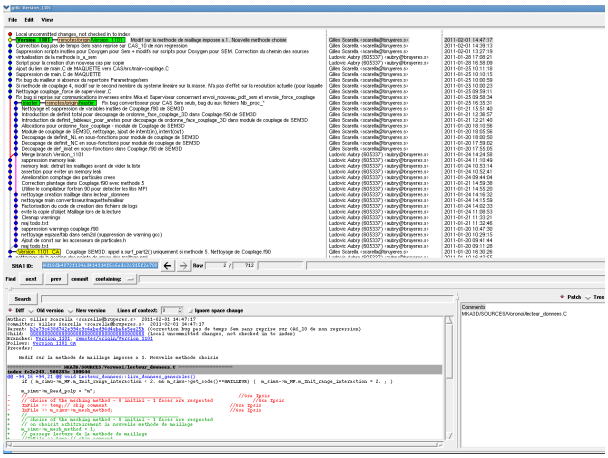

Figure 6.1: Visualisation avec gitk

- Pour configurer ses paramètres (ce qui modifie le fichier ~/.gitconfig)

```
git config --global user.name "toto"
git config --global user.email "toto@example.com"
```

- Sur un terminal couleur, on peut activer les sorties git en couleur

```
git config --global color.ui auto
```

6.2 Démarrage rapide

Un exemple minimal d'utilisation de **git** tient en quelques lignes

```
mkdir exemple
cd exemple
git init
cat > test.txt
Hello world
^D
git add test.txt
git commit -m "Premiere revision"
```

Pour aller plus loin il faut regarder les commandes interactives :

- `git gui` : permet de faire des commits facilement
- `gitk` : permet de naviguer dans l'historique, faire des recherches, etc...
- `git help` et `git xxx --help` : permet d'obtenir la documentation d'une commande `xxx`

Pour partager des sources avec d'autres développeurs, il faut commencer par faire un clone d'un répertoire **git** :

```
git clone /chemin/du/repertoire_source
```

Ensuite entre deux répertoires on utilise les commandes `git push/pull/fetch`

Si on veut une utilisation qui "ressemble" à l'utilisation de **subversion** il faut créer un *repository* central (*ie* un dépot de sources) avec `git clone --bare`

Donc si on part de l'exemple ci-dessus en supposant que le *repository* initial se trouve dans

```
/home/userA/exemple
```

et que l'on veut que `userA` et `userB` utilisent un repository commun sur le compte `/home/appli`, on peut faire

```
userA$ cd /home/appli
userA$ git clone --bare /home/userA/exemple
```

L'utilisateur `userB` fait

```
userB$ cd /home/userB/my_sources
userB$ git clone /home/appli/exemple.git
```

L'utilisateur `userA` peut supprimer le répertoire exemple et le remplacer avec

```
git clone /home/appli/exemple.git
```

Ce qui a pour effet d'enregistrer `/home/appli/exemple.git` comme *repository* "*remote*" sous le nom "`origin`".

git permet de gérer plusieurs *repository* distants communiquant avec votre *repository* local. Chaque *repository* distant se voit attribuer un mot-clef. Le premier *repository* distant est celui depuis lequel on fait un clonage initial, et par défaut, **git** le nomme `origin`.

Il est possible de gérer les *repository* distants (par exemple en ajouter) avec la commande `git remote add xxx` mais il faut bien lire la doc avant, et comprendre les subtilités des *refspec*.

6.3 Commandes utiles

6.3.1 Récupération des sources dans un répertoire de travail

- Pour cloner un dépôt git dans un répertoire de travail, situé de préférence dans son temp (le répertoire ne doit pas exister)

```
git clone ~mka3d/mka3d.git ~/temp/<repertoire_de_travail>
```

6.3.2 Commandes utiles pour les commit

- Pour voir le statut des fichiers et savoir s'ils sont modifiés par rapport au dernier commit,

```
git status
```

- Pour voir les différences par rapport au dernier commit,

```
git diff
```

- Pour valider des modifications avant un commit ou pour ajouter un fichier ne figurant pas dans le dépôt,

```
git add <fichier>
```

- Pour sélectionner précisément les parties à committer du fichier

```
git add -p <fichier>
```

- Pour annuler une modification non validée par git add et revenir à la version du dernier commit du fichier,:

```
git checkout <fichier>
```

- Pour supprimer un fichier et le supprimer du dépôt,

```
git rm <fichier>
```

- Pour faire un commit avec un message

```
git commit -m  "Modif sur fichier "
```

- Pour modifier le message d'un commit

```
git commit --amend -m " Message "
```

6.3.3 Commandes pour modifier un commit

On peut avoir besoin de modifier un commit : par exemple pour le fusionner avec un autre, pour modifier son message ou pour le découper en plusieur parties.

- Pour modifier un commit, à savoir changer son message ou le fusionner avec un autre, il faut remonter dans l'historique (ici on recule de deux versions)

  ```
  git rebase -i HEAD~2
  ```

  Cette commande va présenter dans un éditeur de texte (variable `EDITOR`) la liste des *commits* sélectionnés avec en regard une action à effecturer (par défaut : `apply`, c'est à dire appliquer le *commit*).

  Si on sauvegarde le fichier sans le modifier, la commande va simplement réappliquer les *commits* dans le même ordre.

  On peut déplacer des lignes, dans ce cas l'ordre d'application change.

  Pour modifier un *commit*, on remplace dans le fichier la commande `apply` par la commande `edit`.

  Pour fusionner un *commit* avec le précédant, on utilise la commande `squash`.

  Pour supprimer un *commit* il suffit de supprimer la ligne qui lui correspond dans le fichier.

  Lorsqu'on sauvegarde et quitte l'éditeur, l'outil va éxecuter les commandes du fichier dans l'ordre.

  Si des *commits* ont la commande `edit`, ou s'ils ne s'appliquent pas correctement à cause d'une édition précédante ou d'un changement d'ordre, l'outil s'arrête et rend la main à l'utilisateur qui peut modifier les fichiers concernés, valider et/ou ajouter des *commits* avec les commandes `git add ...; git commit`.

- Pour annuler complètement le dernier commit

  ```
  git reset --hard HEAD^
  ```

6.3.4 Commandes pour conserver des modifications sans les enregistrer

- Pour conserver les modifications courantes et les réutiliser plus tard, sans faire de commit

  ```
  git stash
  ```

- Pour lister les modifications en attente

  ```
  git stash list
  ```

- Pour récupérer les modifications en attente

  ```
  git stash apply
  ```

- Pour supprimer les modifications en attente

  ```
  git stash drop
  ```

6.3.5 Detection d'un bug

La version courante contient un bug, on connaît une autre version qui ne contient pas ce bug. On peut alors trouver la version qui a introduit ce bug en utilisant `git bisect`

- Pour démarrer la recherche :

```
git bisect
```

- La version courante est propre et ne rencontre pas le bug :

  ```
  git bisect good
  ```

- La version courante est buggée :

  ```
  git bisect bad
  ```

  git propose alors une version intermédiaire à tester.

- Pour sauter la version courante dans un tel procédé :

  ```
  git bisect skip
  ```

6.3.6 Gestion des branches

- Pour lister les branches existantes :

  ```
  git branch -a
  ```

- Pour aller sur une branche :

  ```
  git checkout <branche>
  ```

- Pour créer une nouvelle branche :

  ```
  git branch new_branch
  ```

- Pour fusionner la version courante avec la version de la branche distante :

  ```
  git merge
  ```

6.3.7 Récupération des données committées par les autres utilisateurs

- Pour récupérer les modifications distantes sans fusionner :

  ```
  git fetch
  ```

- Pour recaler la branche courante au niveau de la branche distante (permet la mise à jour des fichiers source en tenant compte des modifications distantes) ;

  ```
  git rebase origin/master
  ```

- Pour recaler la branche courante au niveau d'une branche distante qui n'est pas `origin/master` :

  ```
  git rebase origin/Version_1101
  ```

- Pour récupérer les commits distants et faire la fusion avec la version courante :

  ```
  git pull
  ```

6.3.8 Transmission de ses commits aux autres utilisateurs

- Pour transmettre (pousser) les modifications aux autres utilisateurs :

  ```
  git push
  ```

- Pour pousser la branche branche_locale sur origin :

  ```
  git push origin branche_locale
  ```

- Pour transmettre les tags :

  ```
  git push --tags
  ```

- Pour mettre une étiquette sur la version courante :

  ```
  git tag nom
  ```

6.3.9 Précautions

Après les avoir poussés, il est compliqué de modifier des commits. Il faut donc utiliser la commande `git push` avec précaution.

Bibliography

[FAC97] Faccioli, E., R. Maggio, R. Paolucci, and A. Quarteroni (1997). 2D and 3D elastic wave propagation by a pseudo-spectral domain decomposition method. *J. Seismol. 1*, 237-251.

[FES05] Festa, G. and J.-P. Vilotte (2005). The Newmark scheme as velocity-stress time-staggering: an efficient PML implementation for spectral element simulations of elastodynamics. *Geophys. J. Int. 161(3)*, 789-812.

[KOM98] 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. Seismol. Soc. AM. 88(2)*, 368-392.

[KOM99] Komatitsch, D. and J. Tromp (1999). Introduction to the spectral-element method for three-dimensional seismic wave propagation. *Geophys. J. Int. 139(3)*, 806-822.

[PRI94] Priolo, E., J. M. Carcione, and G. Seriani (1994). Numerical simulation of interface waves by high-order spectral modeling techniques. *J. Acoust. Soc. AM. 95(2)*, 681-693.

[SER98] Seriani, G. (1998). 3D large-scale wave propagation modeling by a spectral element method on a Cray T3E multiprocessor. *Computer Methods in Applied Mechanics and Engineering 164(1)*, 235-247.

Index