	<p style="color: red;">Si besoin, timbre DR</p>	<p>Chrono diffusion (code barre) :</p>
	<p style="color: red;">Ne pas diffuser sans autorisation de l'émetteur (à indiquer en cas de diffusion DR)</p>	
	<p>Type de document</p>	<p>Page 1/25</p>

Direction de l'Energie Nucléaire
 Direction déléguée des Activités Nucléaires de Saclay
 Département des Matériaux pour le Nucléaire
 Service de Recherches Métallurgiques Appliquées
 Laboratoire d'étude du Comportement Mécanique des Matériaux

Titre du document


Auteur(s)

DEN/DANS/DMN/SRMA/LC2M/NT ou ST/201x-xxxx/indice

Document émis dans le cadre de l'accord
 XXXXX (si vous l'avez)

Commissariat à l'énergie atomique et aux énergies alternatives
 Centre de Saclay – DMN/SRMA/LC2M – 455 – PC49
 Tél. : 33 – 01 69 08 55 40 – Fax : 33 – 01 69 08 71 67 – laetitia.nicolas@cea.fr
 Etablissement public à caractère industriel et commercial
 R.C.S. PARIS B 775 685 019



	Si besoin, timbre DR Titre du document	Type de document	Page 2/25
	Ne pas diffuser sans autorisation de l'émetteur (à indiquer en cas de diffusion DR)	Référence du document	
		Date : xx/xx/xxxx	Indice : A


NIVEAU DE CONFIDENTIALITE				
DO	DR	CCEA	CD	SD

PARTENAIRES/CLIENTS	ACCORD	TYPE D'ACTION

REFERENCES INTERNES CEA			
DIRECTION D'OBJECTIFS	DOMAINE	PROJET	EOTP
JALON	INTITULE DU JALON	DELAI CONTRACTUEL DE CONFIDENTIALITE	CAHIERS DE LABORATOIRE


SUIVI DES VERSIONS			
INDICE	DATE	NATURE DE L'EVOLUTION	PAGES, CHAPITRES
A	Aout 2012	Emission initiale	

	NOM	FONCTION	VISA
REDACTEUR			
VERIFICATEUR			
VERIFICATEUR			
EMETTEUR			

	Si besoin, timbre DR	Type de document	Page 3/25
	Titre du document	Référence du document	
	Ne pas diffuser sans autorisation de l'émetteur (à indiquer en cas de diffusion DR)	<u>Date</u> : xx/xx/xxxx	<u>Indice</u> : A


MOTS CLEFS**RESUME / CONCLUSIONS** de même niveau de confidentialité que le document

RESUME / CONCLUSIONS rédigé pour être de niveau DO si le document est de niveau DR, CCEA, CD ou SD

	Si besoin, timbre DR	Type de document	Page 4/25
	Titre du document	Référence du document	
	Ne pas diffuser sans autorisation de l'émetteur (à indiquer en cas de diffusion DR)	<u>Date</u> : xx/xx/xxxx	<u>Indice</u> : A


DIFFUSION INITIALE

Destinataires :

	Si besoin, timbre DR Titre du document Ne pas diffuser sans autorisation de l'émetteur (à indiquer en cas de diffusion DR)	Type de document	Page 5/25
		Référence du document	
		Date : xx/xx/xxxx	Indice : A

SOMMAIRE

1.	Présentation générale	6
2.	Installation, exécution et résultats	6
a.	Prérequis.....	6
b.	Installation de 2decomp&fft.....	6
c.	Installation de FoX	7
d.	Installation d'Amitex_fftp.....	7
e.	Exécution	8
f.	Résultats	9
g.	Estimation mémoire	10
3.	Manuel Utilisateur	11
a.	Paramètres d'algorithme.....	11
b.	Chargement	11
c.	Propriétés des matériaux.....	13
d.	Géométrie	15
e.	Utilisation avancée	16
i.	Introduction d'une loi de comportement.....	17
ii.	Modification des sorties	17
4.	Manuel de programmation	18
a.	Règles de programmation	18
b.	Architecture générale du code	18
c.	Fichier amitex_fftp.....	19
d.	Description du module « Material »	20
e.	Description du module « Loading »	20
f.	Description du module « Param_algo »	22
g.	Description du module « Green »	22
h.	Description du module « Io_amitex »	22
i.	Description du module « Error »	23
5.	Annexe 1 Fichiers et lois de comportement	23
6.	Annexe 2 Estimation mémoire affinée.....	24
	Table des codes.....	25
	Table des figures	25
	Bibliographie.....	25

	<p style="color: red; text-align: center;">Si besoin, timbre DR</p> <p style="text-align: center;">Titre du document</p> <p style="color: red; text-align: center;">Ne pas diffuser sans autorisation de l'émetteur (à indiquer en cas de diffusion DR)</p>	Type de document	Page 6/25
		Référence du document	
		Date : xx/xx/xxxx	Indice : A

1. Présentation générale

//LG

Dans la suite, on appellera « cellule » l'assemblage composite sur lequel on effectue les calculs. Le terme « matériau » désigne un composant de la cellule. Un « matériau » est associé à une loi de comportement. On appelle « voxel » un point de la cellule.

2. Installation, exécution et résultats

Le programme Amitex_fftp est codé avec le langage Fortran [1] et la bibliothèque MPI [2], il s'appuie sur deux bibliothèques :

- Decomp2d&fft [3] donne accès à une interface pour la décomposition des données et l'utilisation de Transformée de Fourier Rapide
- FoX [4] permet de lire des fichiers « xml » plus aisément.

a. Prérequis

- Compilateur Fortran (minimum : Intel 13.0.1 ou GNU 4.7.2)
- Bibliothèque MPI
- Bibliothèque fft (fftw [5])

Le programme utilise le reprofilage de tableau de rang supérieur à 1 via un pointeur, introduit par Fortran2008, d'où provient la restriction sur le compilateur.

Le programme a été testé sur les clusters Poincare avec :

- intel 13.0.1 et intelmpi 4.0.3
- gnu 4.7.2 et openmpi 1.6.3

Dans les deux cas on utilise les bibliothèques

- fftw 3.3.3
- 2decomp_fft 1.5.847
- FoX 4.1.2

et Maldives avec :

- gnu 4.8.2 et openmpi 1.8
- fftw 3.3.3
- 2decomp_fft 1.xxx
- FoX 4.xxx


//vérifier les versions utilisées sur maldives:

On notera que les performances sont meilleures avec le compilateur intel.

L'exécution (sur 16 processus) du cas test « polyxCC » nécessite environ 450 secondes avec intel et environ 2000 secondes avec gnu.

La différence apparait lors de l'appel à la fonction « UMAT ».

b. Installation de 2decomp&fft

	<p style="text-align: center;">Si besoin, timbre DR</p> <p style="text-align: center;">Titre du document</p> <p style="text-align: center;">Ne pas diffuser sans autorisation de l'émetteur (à indiquer en cas de diffusion DR)</p>	Type de document	Page 7/25
		Référence du document	
		Date : xx/xx/xxxx	Indice : A

On trouve sur le site de la bibliothèque Decomp2d&fft (voir [3]) l'ensemble des instructions pour l'installation et l'utilisation de la bibliothèque. Ci-dessous une description succincte pour l'installation.

- Décompresser l'archive 2decomp_fftp-xxx.tar
- Ajuster le fichier Makefile.inc (à partir du Makefile.inc.x86) du dossier « src » :
 - Si on souhaite travailler en double précision : « OPTIONS=-DDOUBLE_PREC »
 - Définir la bibliothèque utilisée et le chemin vers celle-ci avec les variables « FFT » et « FFTW_PATH »
 - Commenter l'option « CRAYPTR »
- Exécuter la commande « make » dans le dossier « src »

c. Installation de FoX

Amitex_fftp a été développé et testé avec la version FoX-4.1.2 dont on peut trouver les sources sur la page de téléchargement www1.gly.bris.ac.uk/~walker/FoX/source/. Les instructions de compilation se trouvent à l'adresse www1.gly.bris.ac.uk/~walker/FoX/DoX/Compilation. Ci-dessous une description succincte pour l'installation.

- Décompresser l'archive FoX-xxx.tar
- Dans le dossier créé :
 - exécuter la commande « configure »
 - vérifier dans le fichier arch.make que la variable FC a pour valeur un compilateur fortran satisfaisant le prérequis.
 - puis exécuter la commande « make »

d. Installation d'Amitex_fftp


L'installation d'Amitex_fftp dépend de celles des bibliothèques FoX et 2decomp&fft

- Décompresser l'archive amitex_fftp-xxx.tar
- Ajuster le fichier Makefile du répertoire « src » :
 - Le chemin vers 2decomp&fft avec la variable « IDecomp »
 - Le chemin vers FoX avec la variable « IFoX »
- Exécuter la commande « make » dans le dossier « src »
 - Le dossier « src » doit contenir les fichiers objets (.o) correspondant à chaque fichier source (.f90 ou .F).
 - Le dossier « src/matériaux » doit contenir les fichiers objets (.o) correspondant à chaque fichier source (.f90 ou .F).
 - Le dossier « lib » doit contenir la bibliothèque « libamitex.a ».
 - Le dossier « include » doit contenir les fichiers « .mod ».

Afin de créer un exécutable, l'utilisateur devra, dans le dossier de son choix, copier le Makefile du dossier « exemples » d'Amitex et renseigner les chemins vers :

- la bibliothèque 2decomp&fft avec la variable « IDecomp »
- la bibliothèque FoX avec la variable « IFoX »
- la bibliothèque Amitex avec la variable « IAmitex »

Si l'utilisateur souhaite ajouter des lois de comportement et/ou modifier la fonction « UMAT » il doit alors spécifier dans le Makefile le dossier contenant les sources utilisées via la variable « behavior ». Pour modifier la fonction de sortie « standard » il faut de même indiquer le dossier contenant les sources à utiliser dans le Makefile avec la variable « output ».

	<p style="text-align: center;">Si besoin, timbre DR</p> <p style="text-align: center;">Titre du document</p> <p style="text-align: center;">Ne pas diffuser sans autorisation de l'émetteur (à indiquer en cas de diffusion DR)</p>	Type de document	Page 8/25
		Référence du document	
		Date : xx/xx/xxxx	Indice : A

Dans le cas où l'on utilise uniquement les fonctions proposées par Amitex, les variables « behavior » et « output » doivent être vides.

Ces derniers points sont détaillés dans la partie 3.e.

Remarque :

S'ils sont présents, les dossiers de comportement et de sortie « standard » doivent contenir un Makefile dont des exemples sont fournis dans les dossiers « exemples/comportement » et « exemples/sortie_std ».

Une fois ces variables renseignées, il suffit d'utiliser la commande « make », un exécutable sera créé dans le dossier courant.

Deux cas tests sont disponibles :

- le cas test « beton_fluage » où l'on n'utilise que des fonctions proposées par Amitex
- le cas test « polyxCC » où l'on ajoute des lois de comportement

Dans les deux cas, pour obtenir un exécutable, il suffit de renseigner les chemins vers les bibliothèques dans le Makefile du dossier concerné et d'exécuter la commande « make » dans el dossier contenant le cas test (« exemples/beton_fluage » ou « exemples/polyxCC »).

e. Exécution

L'exécution est lancée par la ligne de commande :

Code 1 Ligne de commande

```
mpirun -np nbproc ./amitex_fftp5 -a algo.xml -c char.xml -m mat.xml -g geom.xml [-s sortie]
```

« nbproc » correspond au nombre de processus utilisés.

« amitex_fftp5 » est le nom de l'exécutable (pour le cas test il s'agit de « amitex_fftp5 »).

Le programme nécessite en entrée des fichiers au format xml qui permettent de définir :

- les paramètres de l'algorithme (-a algo.xml)
- le chargement (-c char.xml)
- les propriétés des matériaux (-m mat.xml)
- la géométrie de la cellule (-g geom.xml)

On peut aussi spécifier le radical du nom des fichiers créés lors de l'exécution du programme avec : « -s radical » ainsi tous le nom de chaque fichier créés au cours de l'exécution commencera par « radical » (par défaut, ce radical est : « sortie »).

Le contenu de ces fichiers sera détaillé plus tard, dans la partie 3 (Manuel utilisateur).

Les deux cas tests contiennent les fichiers nécessaires à leur exécution et l'exécution sera lancée par les lignes de commandes :


Code 2 Exécution des cas tests

Cas test : « beton_fluage »

```
mpirun -np nbproc ./amitex_fftp5 -g geom_beton.xml -m mat_beton.xml -c char_beton.xml -a algo_beton.xml
```

Cas test « polyxCC »

```
mpirun -np nbproc ./amitex_fftp5 -g geom_polyx.xml -m mat_polyx.xml -c char_polyx.xml -a algo_polyx.xml
```


	Si besoin, timbre DR Titre du document Ne pas diffuser sans autorisation de l'émetteur (à indiquer en cas de diffusion DR)	Type de document	Page 9/25
		Référence du document	
		Date : xx/xx/xxxx	Indice : A

On présente ci-dessous des exemples de scripts permettant d'exécuter le programme sur différents clusters.

Code 3 Script Poincare

```
#!/bin/bash
#@ class                = clallmds
#@ job_name              = amitex                # nom du job a executer
#@ total_tasks           = nbproc                #nombre de processus utilisés
#@ node                  = nb_noeuds             #nombre de nœuds
#@ wall_clock_limit      = 00:10:00             #temps maximal de l'execution (h:m:s)
#@ output                = $(job_name).$(jobid).log #nom des fichiers de sortie du terminal
#@ error                 = $(job_name).$(jobid).err # et d'erreur
#@ job_type              = mpich
#@ environment           = COPY_ALL              # permet de reprendre les modules
                                                    # charges dans la session courante
#@ node_usage            = not_shared            # les nœuds utilisés sont consacrés
                                                    #uniquement au job

#@ queue
cd /repertoire # chemin vers le dossier contenant l'executable
mpirun -np nbproc ./amitex_fftp5 -a algo.xml -c char.xml -m mat.xml -g geom.xml [-s sortie]
```

Ce script est lancé en ligne de commande par « lsubmit script »

Code 4 Script Maldives

```
#!/bin/bash
#PBS -S /bin/bash
#PBS -N amitex                # nom du job a executer
#PBS -l nodes=1:ppn=12        # nombre de nœuds, nombre de processus par nœuds
#PBS -l walltime=0:10:0       # temps maximal de l'execution (h:m:s)
cd $PBS_O_WORKDIR             # chemin vers le dossier contenant l'executable

mpirun -np nbproc ./amitex_fftp5 -a algo.xml -c char.xml -m mat.xml -g geom.xml [-s sortie]
```


Ce script est lancé en ligne de commande par « qsub script »

Code 5 Script Airain

```
#!/bin/bash
#MSUB -r amitex                # nom du job a executer
#MSUB -N nb_noeuds            # nombre de nœuds
#MSUB -c nbproc               # nombre de processus utilisés
#MSUB -T 600                  # temps maximal de l'exécution (s)
#MSUB -o ./%l.log             # nom des fichiers de sortie du terminal
#MSUB -e ./%l.err             # et d'erreur. %l est l'identifiant du job
#MSUB -q standard
set -x                        # les nœuds utilisés sont consacrés
                              #uniquement au job
cd ${BRIDGE_MSUB_PWD}         # chemin vers le dossier contenant l'executable
mpirun -np nbproc ./amitex_fftp5 -a algo.xml -c char.xml -m mat.xml -g geom.xml [-s sortie]
```

Ce script est lancé en ligne de commande par « ccc_msub script »

f. Résultats

	Si besoin, timbre DR Titre du document Ne pas diffuser sans autorisation de l'émetteur (à indiquer en cas de diffusion DR)	Type de document	Page 10/25
		Référence du document	
		Date : xx/xx/xxxx	Indice : A

L'exécution fournit trois types de résultats différents :

- La sortie du terminal : toutes les dix itérations et à la fin de chaque pas de chargement, on affiche le pas de temps auquel on se trouve, le dernier critère d'équilibre calculé et temps d'exécution des principales fonctions (les FFT, IFFT, le comportement du matériau (UMAT), l'application du tenseur de Green, l'évaluation du critère d'équilibre et l'écriture des fichiers). Pour une exécution sur un cluster, cette sortie est en générale redirigé vers des fichiers (« .log » ou « .o » par exemple).
- La sortie dite « standard » : à la fin de chaque pas de chargement, on écrit dans le fichier « sortie.std » les moyennes et écart types des contraintes et des déformations dans chaque zone, chaque matériau et sur la cellule entière.
- Des fichiers au format vtk. Le nombre et le contenu de ces fichiers est déterminé dans le fichier d'entrée de chargement (char_beton.xml ou char_polyx.xml par exemple). Ces fichiers auront pour noms : « sortie_x_type.vtk » où :
 - « sortie » peut être renommé (en utilisant le « -s » de la ligne de commande)
 - « x » correspond au numéro du pas de chargement
 - « type » est le type de données extraites, il peut s'agir du champ de contrainte « sig » du champ de déformations « def » où des variables internes « VarInt »

Les dossiers des cas tests contiennent chacun un dossier « resultat » comportant les fichiers résultats d'une exécution du programme en double précision sur le cluster Poincare avec le compilateur intel et les bibliothèques intelmpi et fftw. Les sorties du terminal de ces exécutions ont été redirigées vers les fichiers « .log » et « .err ».

g. Estimation mémoire

Notons n_x, n_y, n_z les dimensions de la cellule. ER et EF sont les nombres d'éléments dans les espaces réels et de Fourier. En supposant l'espace décomposé en « l » lignes et « c » colonnes (on utilise alors $n_{proc} = l \cdot c$ processus), on peut approcher ces valeurs par :

$$ER = n_x \cdot \left\lceil \frac{n_y}{l} \right\rceil \cdot \left\lceil \frac{n_z}{c} \right\rceil \quad (1)$$

$$EF = \left\lceil \frac{n_x}{2} \right\rceil \cdot \left\lceil \frac{n_y}{l} \right\rceil \cdot \left\lceil \frac{n_z}{c} \right\rceil \quad (2)$$

Les symboles $\lceil \cdot \rceil$ représentent une partie entière supérieure.


Notons R le nombre d'octets nécessaire pour représenter un réel (4 octets en simple précision et 8 octets en double précision).

Si « M » représente l'espace mémoire nécessaire à l'exécution du programme, on peut approximer :

$$M = (33 \cdot ER + 31 \cdot EF) \cdot R + M_{cell} \quad (3)$$

Le terme $(33 \cdot ER + 31 \cdot EF) \cdot R$ prend en compte les besoins mémoire des tableaux de contraintes et déformations du programme principal ainsi que ceux de 2decomp&fft.

« M_{cell} » représente l'espace mémoire nécessaire pour représenter la cellule. On considère seulement la variable MattotP dans la mesure où l'empreinte mémoire des structures de chargement, des paramètres d'algorithme et des paramètres d'extraction n'est pas représentative.

	<p style="text-align: center;">Si besoin, timbre DR</p> <p style="text-align: center;">Titre du document</p> <p style="text-align: center;">Ne pas diffuser sans autorisation de l'émetteur (à indiquer en cas de diffusion DR)</p>	Type de document	Page 11/25
		Référence du document	
		Date : xx/xx/xxxx	Indice : A

Pour un matériau donné, si on note $nPos$ le nombre de voxels occupés par ce matériau, $nZone$ le nombre de Zone du matériau, $nCoeff$ son nombre de coefficients et nVI celui des variables internes alors on peut estimer l'empreinte mémoire (en octets) à :

$$M_{mat} = 8. (nPos + 2. nZone) + R. (2nPos. nVI + nZone. nCoeff) \quad (4)$$

L'empreinte mémoire nécessaire pour représenter la cellule « M_cell » sera donc la somme de « M_mat » (pour chaque matériau).

L'approximation de « M_cell » ci-dessus suppose que la répartition des matériaux est assez hétérogène au sein de la cellule.

Dans le cas contraire, se référer à l'Annexe 2 Estimation mémoire affinée.

3. Manuel Utilisateur

Cette partie décrit les différents fichiers permettant de définir complètement la cellule (géométrie et paramètres de matériaux), le chargement, les paramètres de l'algorithme et les données extraites au cours de l'exécution. On y évoque aussi la possibilité d'ajouter des lois de comportement et de modifier la fonction de sortie « standard ».

a. Paramètres d'algorithme

Le fichier de paramètres d'algorithme (préfixé « algo » dans les exemples) permet de définir les différents paramètres de l'algorithme. En l'état actuel, un seul algorithme est disponible (« Basic_scheme ») et le seul paramètre utilisé est la tolérance sur le critère d'équilibre. Des améliorations sont prévues telles que l'accélération de convergence ou la possibilité d'utiliser un opérateur de Green filtré, ces paramètres seront définis dans ce fichier.

Code 6 Exemple de fichier de paramètres d'algorithme

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <Algorithm_Parameters>
3   <Basic_Scheme Convergence_Criterion="1.e-5"/>
4 </Algorithm_Parameters>
5
```


La tolérance sur le critère d'équilibre est ici fixée à 1.e-5.

b. Chargement

Le fichier de chargement (préfixé « char » dans les exemples) permet de définir le chargement souhaité, les données à extraire et les pas de temps auxquels les extraire.

Les données à extraire sont définies pour tous les chargements dans le nœud « Output ». On y indique si on souhaite extraire les contraintes (stress) et/ou les déformations (strain), et pour chaque matériau, les numéros des variables internes à extraire (Code 7, ligne 4-10)

Pour chaque chargement (nœud « Loading »), la discrétisation du temps est définie par le nœud « Time_Discretization » où l'on spécifie le nombre de pas de chargement (attribut « Nincr ») et le type

	Si besoin, timbre DR	Type de document	Page 12/25
	Titre du document	Référence du document	
	Ne pas diffuser sans autorisation de l'émetteur (à indiquer en cas de diffusion DR)	<u>Date</u> : xx/xx/xxxx	<u>Indice</u> : A


de discrétisation du temps (attribut « Discretization ») qui peut être linéaire (« Linear ») ou définie par l'utilisateur (« User »). Si la discrétisation est linéaire (Code 7, ligne 33), on précise le temps final du chargement (« Tfinal »), sinon (Code 7, ligne 16-20), l'utilisateur donne la liste des temps en fin de pas de chargement (nœud « Time_List »).

Le temps initial d'un chargement est 0 s'il s'agit du premier chargement et le temps final du chargement précédent sinon.

On définit, pour chaque composantes (nœuds « xx », « yy », « zz », « xy », « xz », « yz ») le pilotage du chargement (attribut « Driving ») : en contrainte (« Stress ») ou en déformation (« Strain »). Le pilotage en contraintes n'est cependant pas encore implémenté. On indique aussi le type d'évolution du chargement (« Evolution ») qui peut être constant (« Constant ») ou linéaire (« Linear »). Dans le cas d'une évolution linéaire, on précise la valeur finale du chargement (« Value »). La valeur initiale du chargement est la valeur moyenne de la contrainte ou de la déformation (selon le type de pilotage) obtenue au chargement précédent. S'il s'agit du premier chargement, cette valeur est nulle.

Les paramètres de pilotages de l'exemple Code 7 se trouvent aux lignes 24 à 29 et 37 à 42.

Enfin, on indique les pas de chargement auxquels on doit extraire les données souhaitées (nœud « Output_Discretization »).

	Si besoin, timbre DR Titre du document Ne pas diffuser sans autorisation de l'émetteur (à indiquer en cas de diffusion DR)	Type de document	Page 13/25
		Référence du document	
		Date : xx/xx/xxxx	Indice : A


Code 7 Exemple de fichier de chargement

1	<?xml version="1.0" encoding="UTF-8"?>
2	<Loading_Output>
3	<!-- OUPUT QUANTITIES -->
4	<Output>
5	<!-- extraction des contraintes (stress= "1") -->
6	<StressStrain Strain = "0" Stress = "1"/>
7	<!-- liste des variables internes a extraire pour chaque materiau-->
8	<IntVar numM="1" List=""/>
9	<IntVar numM="2" List="1 3 8"/>
10	</Output>
11	
12	<!-- SUCCESSIVE LOADINGS AND OUTPUT TIMES -->
13	<!-- noeuds de chargement -->
14	<Loading Tag="1">
15	<!--discretisation du temps definie par l'utilisateur sur 10 increment -->
16	<Time_Discretization Discretization="User" Nincr="10" />
17	<!-- temps des increments (definis par l'utilisateur) -->
18	<Time_List>
19	32832 83808 143424 211680 289440 380160 483840 604800 738720 898560
20	</Time_List>
21	<!-- aucune extraction pour ce chargement -->
22	<Output_Discretization List=""/>
23	<!-- le pilotage se fait lineairement, en deformation pour toute les composantes -->
24	<xx Driving="Strain" Evolution="Linear" Value="-5e-4"/>
25	<yy Driving="Strain" Evolution="Linear" Value="1.5e-4" />
26	<zz Driving="Strain" Evolution="Linear" Value="1.5e-4" />
27	<xy Driving="Strain" Evolution="Linear" Value="0" />
28	<xz Driving="Strain" Evolution="Linear" Value="0" />
29	<yz Driving="Strain" Evolution="Linear" Value="0" />
30	</Loading>
31	<Loading Tag="2">
32	<!-- discretization lineaire du temps -->
33	<Time_Discretization Discretization="Linear" Nincr="22" Tfinal="25920000" />
34	<!-- extraction du dernier pas de chargement -->
35	<Output_Discretization List="22"/>
36	<!-- le chargement est constant, en contrainte -->
37	<xx Driving="Stress" Evolution="Constant" />
38	<yy Driving="Stress" Evolution="Constant" />
39	<zz Driving="Stress" Evolution="Constant" />
40	<xy Driving="Stress" Evolution="Constant" />
41	<xz Driving="Stress" Evolution="Constant" />
42	<yz Driving="Stress" Evolution="Constant" />
43	</Loading>
44	</Loading_Output>

On rappelle que le pilotage en contrainte n'est pas encore implémenté.

c. Propriétés des matériaux

Ce fichier (préfixé « mat » dans les exemples) permet de décrire les propriétés du matériau de référence et des matériaux de la cellule.

	Si besoin, timbre DR	Type de document	Page 14/25
	Titre du document	Référence du document	
	Ne pas diffuser sans autorisation de l'émetteur (à indiquer en cas de diffusion DR)	<u>Date</u> : xx/xx/xxxx	<u>Indice</u> : A

Les propriétés du matériau de référence sont les coefficients de Lamé (« lambda0 » et « mu0 »), exemple : Code 8, ligne 3.

Pour déterminer le comportement mécanique en un point de la cellule (voxel) on doit pouvoir identifier la loi de comportement qui s'y applique, les coefficients et variables internes en ce point.

Un matériau (identifié par un numéro de matériau) est constitué d'un ensemble de voxels régis par une même loi de comportement (identifiée par un numéro de loi). Un matériau est constitué d'une ou plusieurs « zones ».


Une « zone » d'un matériau est un sous-ensemble des voxels du matériau partageant les mêmes coefficients. Enfin chaque voxel possède ses propres variables internes.

Les propriétés des matériaux de la cellule regroupent le numéro de matériau (attribut « numM »), le numéro de loi associé (attribut « Law_Number »), la valeur des coefficients et des variables internes (Code 8, lignes 7 à 11 et 15 à 33).

Les coefficients (nœud « Coeff ») et variables internes (nœud « IntVar ») sont identifiés par leurs indices et peuvent être définis comme :

- « Constant », la valeur (« Value ») donnée à ce coefficient (ou variable interne) sera constante pour l'ensemble du matériau.
- « Constant_zone », le coefficient (ou variable interne) se voit affecter une valeur selon la zone du matériau.

Les coefficients (variables internes) constants par zone peuvent être définis directement dans le fichier xml (Code 8, lignes 17 à 22) ou dans un autre fichier xml (Code 8, ligne 24). Dans le cas où on utilise un fichier xml pour définir les valeurs des coefficients (variables internes), l'ordre des valeurs est important : la première valeur correspond à la zone numéro « 1 » et ainsi de suite.

	Si besoin, timbre DR Titre du document Ne pas diffuser sans autorisation de l'émetteur (à indiquer en cas de diffusion DR)	Type de document	Page 15/25
		Référence du document	
		Date : xx/xx/xxxx	Indice : A

Code 8 Exemple de fichier des propriétés matériaux

1	<Materials>
2	<!-- REFERENCE MATERIAL -->
3	<Reference_Material Lambda0="2.0952e+10" Mu0="1.5014e+10"/>
4	
5	<!-- MATERIAL 2 -->
6	<!-- numeros de materiau et de loi -->
7	<Material numM="2" Law_Number="1" >
8	<!-- deux coefficients constant -->
9	<Coeff Index="1" Type="Constant" Value="4.0385e+10"/>
10	<Coeff Index="2" Type="Constant" Value="2.6923e+10"/>
11	</Material>
12	
13	<!-- MATERIAL 1 -->
14	<!-- numeros de materiau et de loi -->
15	<Material numM="1" Law_Number="40">
16	<!-- coefficient 1: constant par zone -->
17	<Coeff Index="1" Type="Constant_Zone">
18	<!-- valeur du coefficient 1 pour la zone 1 -->
19	<Zone numZ="1" Value="5.235e+10" />
20	<!-- valeur du coefficient 1 pour la zone 2 -->
21	<Zone numZ="2" Value="3.912e+10" />
22	</Coeff>
23	<!-- coefficient 2 constant par zone defini par un fichier xml -->
24	<Coeff Index="2" Type="Constant_Zone " File="Coeff2.xml"/>
25	
26	<!-- les variables internes sont definies de la meme facon-->
27	<IntVar Index="1" Type="Constant" Value="0."/>
28	<Coeff Index="2" Type="Constant_Zone">
29	<Zone numZ="1" Value="5.235e+10" />
30	<Zone numZ="2" Value="3.912e+10" />
31	</Coeff>
32	<Coeff Index="3" Type="Constant_Zone " File="Var3.xml"/>
33	</Material>
34	</Materials>


Code 9 Exemple de fichier de coefficients

1	<?xml version="1.0" encoding="UTF-8"?>
2	<Coeff2 nZone="2">
3	1.429430417
4	3.905850691
5	</Coeff2>

On rappelle que dans ces fichiers, l'ordre des valeurs est important et correspond au numéro de zones.

d. Géométrie

Le fichier de géométrie (préfixé « geom » dans les exemples) permet de définir la géométrie de la cellule : on associe à chaque voxel de la cellule un numéro de matériau et un numéro de zone.

	<p style="text-align: center;">Si besoin, timbre DR</p> <p style="text-align: center;">Titre du document</p> <p style="text-align: center;">Ne pas diffuser sans autorisation de l'émetteur (à indiquer en cas de diffusion DR)</p>	Type de document	Page 16/25
		Référence du document	
		Date : xx/xx/xxxx	Indice : A

Ces numéros sont introduits via des fichiers vtk. Le fichier des numéros de matériaux est indiqué par l'attribut « numM », celui des zones par l'attribut « numZ ».

Code 10 Exemple de fichier de géométrie

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- vtk files to define geometry -->
3 <Geometry numM="material.vtk" numZ="zone.vtk" />

```

Un des deux fichiers peut ne pas être spécifié. Dans ce cas, on considère qu'il s'agit d'un champ uniforme et égal à « 1 ».

Les numéros de zones et de matériaux doivent être indexés continument en débutant par « 0 » ou « 1 ».

Restrictions sur les fichiers vtk:

- On utilise le format « simple legacy »:
 - le fichier doit donc contenir un entête de 10 lignes (voir le Code 11).
 - la partie binaire du fichier doit donc être écrite en « big endian ». Avec Matlab par exemple, si on utilise la fonction « fwrite », on donnera à la variable « machineformat » la valeur 'b'.
- Les fichiers vtk ne doivent contenir qu'un seul champ.
- Les données peuvent être des entiers codés sur:
 - 1 octet (type « char »)
 - 2 octets (type « short » ou « unsigned_short »)
 - 4 octets (type « int » ou « unsigned_int »)
 - 8 octets (type « long » ou « unsigned_long »)
- le cas de données non signées (« unsigned... ») la valeur maximale reste limitée à celle du type signé correspondant (i.e. le maximum pour un « unsigned_int » est le même que le maximum pour un "int" etc.).

Code 11 Exemple d'entête de fichier vtk

```


1 # vtk DataFile Version 4.5
2 Matériau
3 BINARY
4 DATASET STRUCTURED_POINTS
5 DIMENSIONS 66 66 66
6 ORIGIN 0.000 0.000 0.000
7 SPACING 1.000000 1.000000 1.000000
8 CELL_DATA 274625
9 SCALARS MaterialId unsigned_short
10 LOOKUP_TABLE default

```

Important: On doit ajouter 1 à chaque dimension. Dans l'exemple ci-dessus, il s'agit d'une cellule dont les dimensions sont (65, 65, 65).

e. Utilisation avancée

L'utilisateur a la possibilité de changer ou de compléter, selon ces besoins, certaines fonctions d'Amitex_fft. Ces fonctions sont liées au comportement des matériaux et à la sortie « standard ». La marche à suivre pour effectuer ces modifications est expliquée dans cette partie.

	<p align="center">Si besoin, timbre DR</p> <p align="center">Titre du document</p> <p align="center">Ne pas diffuser sans autorisation de l'émetteur (à indiquer en cas de diffusion DR)</p>	Type de document	Page 17/25
		Référence du document	
		Date : xx/xx/xxxx	Indice : A

i. Introduction d'une loi de comportement

L'utilisateur a la possibilité d'introduire ses propres lois de comportement. Pour ce faire, il doit créer un dossier contenant :

- les fichiers sources des lois de comportement ajoutées
- un fichier « umat.F » (ou umat.f90) contenant la fonction « umat » (selon le modèle de CAST3M)
- un makefile

Le chemin vers ce dossier doit alors être indiqué dans le Makefile utilisé pour générer l'exécutable (variable « behavior »).

Les sources des fonctions de comportement et de la fonction umat ainsi qu'un Makefile se trouvent dans le dossier « exemples/comportement ».

Important :

- La fonction « umat » de l'utilisateur doit conserver la même signature, cette dernière est reprise de CAST3M.
- Un fichier de ce dossier ne peut pas avoir le même nom qu'un fichier de loi de comportement déjà utilisé dans Amitex_fftp (voir la liste dans l'Annexe 1)

Remarque :

- A moins qu'il n'y ait un ordre de compilation particulier, le Makefile à ajouter dans le dossier mentionné ci-dessus peut être une copie du Makefile donné en exemple dans le dossier « exemples/comportement ».
- Le dossier « exemples/polyxCC/comportement_umat » constitue un exemple d'introduction de loi de comportement.

ii. Modification des sorties

L'utilisateur a la possibilité de personnaliser le contenu du fichier de sortie standard. Pour ce faire il doit modifier le fichier « sortie_std.f90 ». Celui-ci permet de calculer et d'écrire dans un fichier (à la fin de chaque pas de chargement) des grandeurs d'intérêts telles que les moyennes et écart types des champs de contraintes et de déformations. Il contient deux fonctions : « initSortie_std » et « sortie_std ».

La première est utilisée pour écrire l'entête du fichier « sortie.std » et la seconde permet de calculer et d'écrire lesdites valeurs.

On rappelle que le fichier écrit se nomme « sortie.std » par défaut, il peut être renommé en utilisant l'option « -s radical » en ligne de commande, dans ce cas il sera nommé « radical.std »


Pour récrire ces deux fonctions, l'utilisateur doit créer un dossier contenant :

- les fichiers sources ajoutées
- un makefile

Le chemin vers ce dossier doit alors être indiqué dans le Makefile utilisé pour générer l'exécutable (variable « output »).

Les sources des fonctions « initSortie_std » et « sortie_std » ainsi qu'un Makefile se trouvent dans le dossier « exemples/sortie_std ».

Important :

	<p style="color: red; text-align: center;">Si besoin, timbre DR</p> <p style="text-align: center;">Titre du document</p> <p style="color: red; text-align: center;">Ne pas diffuser sans autorisation de l'émetteur (à indiquer en cas de diffusion DR)</p>	Type de document	Page 18/25
		Référence du document	
		Date : xx/xx/xxxx	Indice : A

- les moyennes sont utilisées pour déterminer les valeurs initiales des chargements. L'utilisateur devra donc être prudent si cette fonction est remplacée.
- Le code source ajouté doit contenir les deux fonctions « initSortie_std » et « sortie_std » et en conserver les signatures.

Remarque :

- A moins qu'il n'y ait un ordre de compilation particulier, le Makefile à ajouter dans le dossier mentionné ci-dessus peut être une copie du Makefile donné en exemple dans le dossier « exemples/sortie_std ».

4. Manuel de programmation

Cette partie s'adresse aux personnes souhaitant comprendre le code en vue de le faire évoluer. On y aborde le contenu des sources, les types dérivés et les fonctions implémentés ainsi que leurs finalités.

a. Règles de programmation

Pour des raisons de précision, les réels doivent être déclarés et initialisés avec les types définis dans la librairie 2decomp : on utilise

- « mytype » pour les déclarations de variables
- « _mytype » pour les affectations de variables
- « real_type » ou « complex_type » pour les types MPI

Code 12 Règles de programmation


```
! Exemple
!déclaration de deux réels
real(mytype) :: crit_eq, criti
!initialisation d'un réel
crit_eq = 0._mytype
utilisation d'un type MPI :
call MPI_ALLREDUCE(criti,crit_eq,1,real_type,MPI_SUM,MPI_COMM_WORLD, ierr)
```

b. Architecture générale du code

Le dossier libAmitex contient trois dossiers :

- Le dossier « src » contient l'ensemble des sources (.f90) et les fichiers objets associés (après compilation).
- Le dossier « include » contient (après compilation) les fichiers « .mod » à inclure lors de l'édition de lien.
- Le dossier « lib » contient (après compilation) la bibliothèque amitex (« libamitex.a »).

Amitex_fftp utilise des types dérivés (structures) pour décrire les matériaux, paramètres d'algorithmes et les données à extraire au cours de l'exécution du programme. Chacune de ces structures est définie dans un module avec les fonctions qui lui sont associées.

	<p align="center">Si besoin, timbre DR</p> <p align="center">Titre du document</p> <p align="center">Ne pas diffuser sans autorisation de l'émetteur (à indiquer en cas de diffusion DR)</p>	Type de document	Page 19/25
		Référence du document	
		Date : xx/xx/xxxx	Indice : A

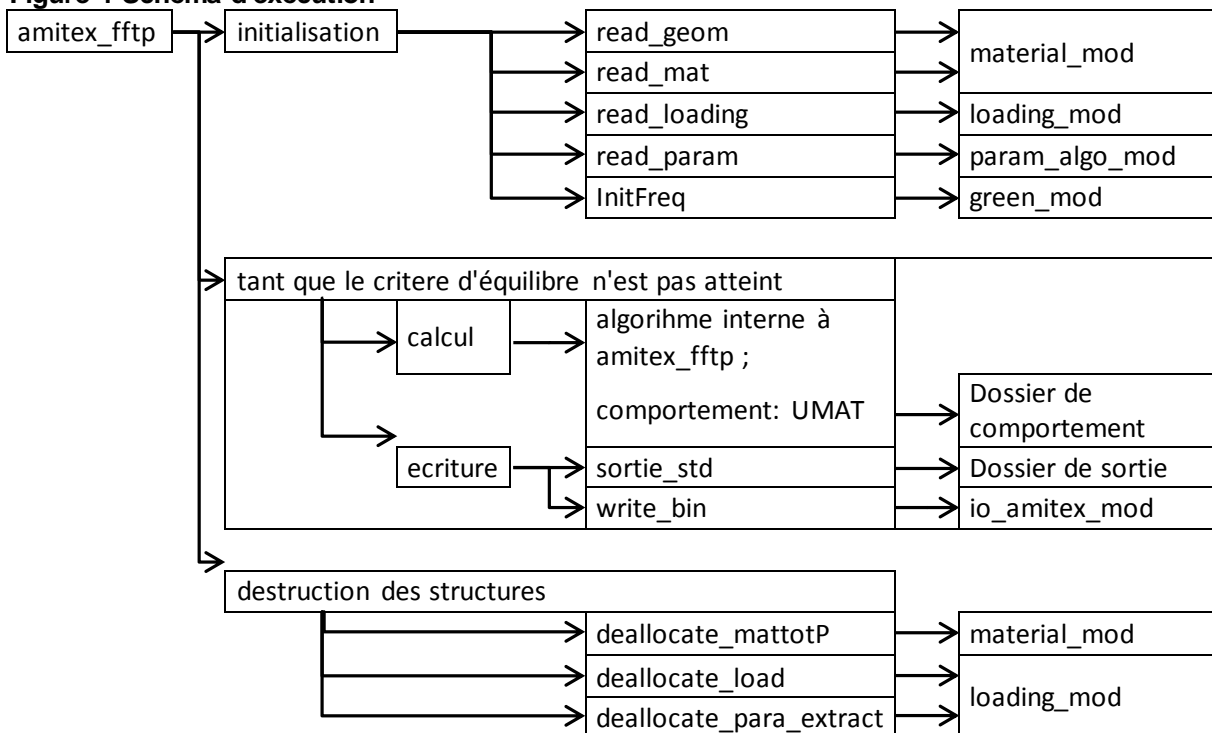
Pour chaque structure, on déclare une variable globale. Toutes les allocations ou dé-allocation de ces variables se font dans le module concerné. On détaille ci-après le contenu des modules et du programme principal.

c. Fichier amitex_fftp

Il s'agit du fichier contenant le programme principal : « amitex_fftp ». Le programme principal est constitué :

- de l'appel aux fonctions de lecture des fichiers xml et vtk : « read_geom », « read_mat », « read_load » et « read_param » (voir la description des modules ci-après).
- de l'algorithme de simulation du comportement. Un seul algorithme est implémenté, celui du « schéma de base » décrit par H. Moulinec & P. Suquet [1].
- de l'écriture des résultats.

Figure 1 Schéma d'exécution




Les transformées de Fourier en trois dimensions sont réalisées à l'aide de la bibliothèque 2decomp&fft. Les fonctions de FFT utilisées prennent en paramètres des tableaux de dimension 3.

Pour réaliser la FFT d'un champ de contrainte ou de déformation, on effectue la FFT de chacune de ses composantes. Ces composantes doivent donc être des tableaux de dimension 3. Or par cohérence avec la structure « MATERIAL » (défini dans la partie 4.d), ces tableaux sont de dimension 1 dans l'espace réel.

Afin d'utiliser les fonctions évoquées ci-dessus, on a recours à un reprofilage par pointeur de ces tableaux.

Ce fichier contient aussi les fonctions :

- lire_commande : lit la ligne de commande (voir le Code 1 Ligne de commande de la partie 2.e)
- set_pRow_pCol : détermine les nombres de lignes et de colonnes de la décomposition.

	<p style="text-align: center;">Si besoin, timbre DR</p> <p style="text-align: center;">Titre du document</p> <p style="text-align: center;">Ne pas diffuser sans autorisation de l'émetteur (à indiquer en cas de diffusion DR)</p>	Type de document	Page 20/25
		Référence du document	
		Date : xx/xx/xxxx	Indice : A

d. Description du module « Material »

Le module « material_mod » est utilisé pour définir la géométrie et les propriétés de la cellule. Pour cela on utilise la variable globale « MattotP », un tableau de structure « MATERIAL ». Chaque élément de la variable « MattotP » est donc une variable de type « MATERIAL ».

Un matériau est identifié par un numéro de matériau (numM), on lui associe une loi de comportement à l'aide d'un numéro de loi (numeLoi). Un matériau contient une ou plusieurs « zone(s) » et chaque zone possède des coefficients qui lui sont propres, ces derniers sont stockés dans le tableau « Coeff ». Les positions des voxels du matériau sont stockés dans le tableau « pos », ordonné par numéro de zone. Enfin à chaque élément du tableau « pos » correspond un élément des tableaux « VarInt » et « VarInt0 » qui sont les variables internes associés au voxel.


Code 13 Structure MATERIAL

1	type MATERIAL	
2	integer	:: NumM !numero du materiau
3	character(len=16)	:: NumeLoi ! numero de loi associe au materiau
4	integer	:: Ncoeff ! nombre de coefficients
5	integer	:: NvarInt ! nombre de variable internes
6		
7	integer(kind=8),allocatable,dimension(:)	:: Pos ! position linearisee des voxels
8		! ordonnes par numero de zone croissant
9	integer(kind=8),allocatable,dimension(:,:):	:: Zone
10		! zone(i,1): indice du dernier element de la zone (dans le tableau pos)
11		! Zone(i,2): numéro de zone non parallelise (pour le post-traitement)
12	real(mytype),allocatable, dimension(:,:)	:: Coeff
13		! coefficients, dimensions: (Ncoeff, nombre de zone du materiau)
14	real(mytype),allocatable, dimension(:,:)	:: VarInt ! variables internes
15	real(mytype),allocatable, dimension(:,:)	:: VarInt0 ! variables internes
16		! Les tableaux de variables internes ont pour dimensions:
17		! le nombre de variable internes du materiau,
18		! le nombre de voxel du materiau dans le pinceau)
19	end type MATERIAL	

Ce module contient les fonctions :

- Read_geom : associe à chaque voxel de la cellule un numéro de matériau et un numéro de zone.
- Read_mat : initialise les propriétés de chaque matériau (numéro de loi, valeurs des coefficients et variables internes) à l'aide du fichier xml des propriétés matériau.
- Behavior : permet le calcul du comportement mécanique de la cellule.
- initParam_umat: initialisation des paramètres de la fonction UMAT.
- deallocate_MattotP: dé-allocation de MattotP.

e. Description du module « Loading »

	<p style="text-align: center;">Si besoin, timbre DR</p> <p style="text-align: center;">Titre du document</p> <p style="text-align: center;">Ne pas diffuser sans autorisation de l'émetteur (à indiquer en cas de diffusion DR)</p>	Type de document	Page 21/25
		Référence du document	
		Date : xx/xx/xxxx	Indice : A

Le module « loading_mod » permet de définir les chargements imposés lors de la simulation avec la structure « LOADING » ainsi que les données à extraire à l'aide de la structure « PARAM_EXTRACT ».

Code 14 structure LOADING

1	type LOADING	
2	integer	:: Nincr !nombre d'increment de temps
3	real(mytype),allocatable,dimension(:)	:: time !temps a chaque increment
4	integer	:: Discretization !type de discretisation du temps
5		
6	integer, dimension(6)	:: driving ! type de pilotage 0: deformation 1: contrainte
7	real(mytype), dimension(6)	:: value ! valeurs imposees en fin de chargement
8	integer, dimension(6)	:: evolution ! type d'evolution du chargement
9	end type LOADING	
10		

Code 15 structure PARAM_EXTRACT

1	type PARAM_EXTRACT	
2	!variables a extraire dans le fichier VTK	
3	integer(kind=1)	:: defVTK ! extraction des deformation=1 (0 sinon)
4	integer(kind=1)	:: sigVTK ! extraction des contraintes=1 (0sinon)
5		
6	integer(kind=1),allocatable,dimension(:)	:: tpsVTK ! temps auxquels extraire les donnees
7		! dimension : nombre total de pas de chargement
8		
9	type(TABLEAU_INT1),allocatable,dimension(:)	:: varVTK ! variables internes a extraire
10		! dimension de VarVTK:nombre de materiaux
11		! dimension de VarVTK(i) : nombre de variables internes du materiau i
12	end type PARAM_EXTRACT	
13		

La structure « PARAM_EXTRACT » utilise un type dérivé : « TABLEAU_INT1 », il s'agit d'un tableau allouable (allocatable) d'entiers de 1 octet pour déclarer la variable « varVTK ». On utilise une telle structure dans la mesure où tous les matériaux n'ont pas le même nombre de variables internes.


Ces structures sont renseignées à l'aide de la fonction :

- read_load: lecture du chargement et des paramètres d'extraction à partir du fichier xml de chargement.

Le fichier xml évoqué ci-dessous réfère au fichier xml de chargement.

Cette fonction s'appuie sur les fonctions suivantes :

- get_loadList_xml: lecture d'un pas de chargement.
- get_loadTime: extrait du fichier xml les informations des temps de chargement.
- get_load: extrait du fichier xml, pour un pas de chargement et une composante les informations relatives au pilotage.
- repartition_discretization: associe un entier au type de discrétisation du temps souhaité (voir les correspondances ci-après).

	<p style="text-align: center;">Si besoin, timbre DR</p> <p style="text-align: center;">Titre du document</p> <p style="text-align: center;">Ne pas diffuser sans autorisation de l'émetteur (à indiquer en cas de diffusion DR)</p>	Type de document	Page 22/25
		Référence du document	
		Date : xx/xx/xxxx	Indice : A

- repartition_evolution: associe un entier au type d'évolution du chargement souhaité (voir les correspondances ci-après).
- set_time: calcul le temps en fin de chaque pas de chargement.

Pour la discrétisation du temps, on associe :

- 0 à une discrétisation définie par l'utilisateur,
- 1 à une discrétisation linéaire.

Pour l'évolution du chargement, on associe :

- 0 à un chargement constant,
- 1 à une évolution linéaire du temps.

Lors de l'exécution, on calcule le chargement imposé à l'aide des fonctions :

- get_curent_loading: calcul le chargement courant.
- get_loading_value: calcul d'une composante du chargement courant.

La dé-allocation des structures se fait avec les fonctions :

- deallocate_load: dé-allocation du chargement.
- deallocate_param_extract: dé-allocation des paramètres d'extraction.

f. Description du module « Param_algo »

Le module « param_algo_mod » définit la structure des paramètres d'algorithme.

Code 16 structure PARAM_ALGO

1	type PARAM_ALGO	
2	real(mytype)	:: tol_criteq ! <i>tolerance sur le critere d'equilibre</i>
3	end type PARAM_ALGO	

La lecture des paramètres de l'algorithme se fait avec la fonction « read_param » (il s'agit de la seule fonction du module).

g. Description du module « Green »


Ce module contient les fonctions liées aux fréquences et à l'opérateur de Green :

- initFreq: initialisation du tableau de fréquences
- apply_green: application du tenseur de Green
- eval_crit_eq: calcul du critère d'équilibre

h. Description du module « lo_amitex »

Ce module regroupe des fonctions liées aux « entrées-sorties » telles que la lecture ou l'écriture de fichier vtk :

- read_header_vtk: lecture de l'entête d'un fichier vtk
- write_header_vtk: écriture de l'entête d'un fichier vtk
- read_bin: lecture de la partie binaire d'un fichier vtk

	<p style="text-align: center;">Si besoin, timbre DR</p> <p style="text-align: center;">Titre du document</p> <p style="text-align: center;">Ne pas diffuser sans autorisation de l'émetteur (à indiquer en cas de diffusion DR)</p>	Type de document	Page 23/25
		Référence du document	
		Date : xx/xx/xxxx	Indice : A

- write_bin: écriture de la partie binaire d'un fichier vtk

Des fonctions d'analyse des données (de parsing) :

- nextInt: recherche de l'entier suivant dans une chaîne de caractères
- nextReal: recherche du réel suivant dans une chaîne de caractères
- get_int_vect: lecture d'un tableau d'entiers dans une chaîne de caractères

Des fonctions relatives à la lecture des fichiers xml :

- getNodeList: rechercher une liste de nœuds dans un fichier xml
- isPresent: vérifier la présence d'un attribut dans un fichier xml
- get_int_xml: lecture d'un entier dans un fichier xml
- get_real_xml: lecture d'un réel dans un fichier xml
- get_str_xml: lecture d'une chaîne de caractère dans un fichier xml
- get_vect_xml: lecture des coefficients ou variables internes du matériau dans un fichier xml
- getCoeffFromFile: lecture de la liste des valeurs d'un coefficient (ou d'une variable interne) dans un fichier xml (cas constant par zone) pour un matériau

i. Description du module « Error »

Ce module est utilisé pour la gestion des erreurs, il contient deux fonctions :

- Amitex_abort : affiche les avertissements et messages d'erreur. Cette fonction interrompt le programme si l'erreur est jugée grave (par le programmeur).
- Check_amitex_abort : interrompt l'exécution si une erreur est détectée précédemment.

5. Annexe 1 Fichiers et lois de comportement

On trouve ci-dessous une liste des fichiers où sont implémentées les lois de comportement.

Pour chaque fichier on indique :

- Le nom de la loi de comportement,
- le numéro de loi auquel elle est associée (par la fonction umat d'Amitex_fftp) et le nom de la fonction,
- Le nom de la fonction associée.

Les types des paramètres utilisés se trouvent en commentaire dans les fichiers « libAmitex/src/umat.F », « exemples/comportement/umat.F » ou dans la documentation de la fonction umat de CAST3M.


La signature des fonctions se trouve dans les fichiers mentionnés ci-dessous dans le répertoire « libAmitex/src/umat.F » ou « exemples/comportement/umat.F ».

- elasiso.f90

loi élastique isotrope
numéro de loi : 1
elasiso

- visoeelas_maxwell.f90

loi viscoélastique de maxwell généralisé
numéro de loi : 40
visoeelas_maxwell

	Si besoin, timbre DR Titre du document Ne pas diffuser sans autorisation de l'émetteur (à indiquer en cas de diffusion DR)	Type de document	Page 24/25
		Référence du document	
		Date : xx/xx/xxxx	Indice : A

6. Annexe 2 Estimation mémoire affinée

La répartition des matériaux au sein de la cellule peut avoir une influence non négligeable sur la mémoire nécessaire à l'exécution du programme du fait de la répartition des données sur les différents processus.

La décomposition est effectuée avec 2decomp.

Une cellule de taille (nx, ny, nz) est décomposée en pinceaux de taille $(nx, \frac{ny}{l}, \frac{nz}{c})$ ou « l » et « c » sont les nombres de lignes et de colonnes de la décomposition (on utilise donc $nbproc = l.c$ processus).

La figure ci-dessous illustre la décomposition d'une cellule de dimension (65, 65, 65) sur 4 lignes et 4 colonnes, en notant « pi » le processus numéro i.

Figure 2 Exemple de décomposition

Lignes	48-65	p12	p13	p14	p15
	32-48	p8	p9	p10	p11
	17-32	p4	p5	p6	p7
	1-16	p0	p1	p2	p3
		1-16	17-32	32-48	48-65
		colonnes			

Une estimation plus fine de la mémoire nécessaire pour la variable « MattotP » que celle proposé dans la partie 2.g consiste à calculer sur chaque processus les besoins mémoires de la variable « MattotP » que l'on note « M_proc_i ».

Pour un matériau donné, notons nPos_i le nombre de voxels occupés par ce matériau sur le processus i, nZone_i le nombre de zones de ce matériau sur le processus i, nCoeff et nVI les nombres de coefficients et de variables internes de ce matériau.

Alors pour le processus i, l'espace mémoire nécessaire pour ce matériau est approché par :

$$M_{mat_i} = 8. (nPos_i + 2. nZone_i) + R. (2nPos_i. nVI + nZone_i. nCoeff) \quad (5)$$

On peut alors approcher « M_proc_i » par la somme des « M_mat_i ».

En supposant que la mémoire disponible pour chaque processus est identique, la mémoire totale nécessaire est donnée par l'équation (3):

$$M = (33. ER + 31. EF). R + M_{cell} \quad (6)$$

Où :

$$M_{cell} = \max_i (M_{proc_i}). nbproc \quad (7)$$


	Si besoin, timbre DR Titre du document Ne pas diffuser sans autorisation de l'émetteur (à indiquer en cas de diffusion DR)	Type de document	Page 25/25
		Référence du document	
		Date : xx/xx/xxxx	Indice : A

TABLE DES CODES

Code 1 Ligne de commande	8
Code 2 Exécution des cas tests	8
Code 3 Script Poincare.....	9
Code 4 Script Maldives.....	9
Code 5 Script Airain.....	9
Code 6 Exemple de fichier de paramètres d'algorithme	11
Code 7 Exemple de fichier de chargement.....	13
Code 8 Exemple de fichier des propriétés matériaux	15
Code 9 Exemple de fichier de coefficients	15
Code 10 Exemple de fichier de géométrie	16
Code 11 Exemple d'entête de fichier vtk	16
Code 12 Règles de programmation	18
Code 13 Structure MATERIAL	20
Code 14 structure LOADING	21
Code 15 structure PARAM_EXTRACT	21
Code 16 structure PARAM_ALGO	22

TABLE DES FIGURES

Figure 1 Schéma d'exécution	19
Figure 2 Exemple de décomposition.....	24

BIBLIOGRAPHIE

- [1] Fortran. (s.d.). Récupéré sur http://www.idris.fr/data/cours/lang/fortran/choix_doc.html
[2] MPI. (s.d.). Récupéré sur http://www.idris.fr/data/cours/parallel/mpi/choix_doc.html
[3] Bibliothèque 2decomp&fft. (s.d.). Récupéré sur <http://www.2decomp.org/>
[4] bibliothèque FoX. (s.d.). Récupéré sur www1.gly.bris.ac.uk/~walker/FoX
[5] Bibliothèque fftw. (s.d.). Récupéré sur <http://www.fftw.org/>