Table des matières

Table des matières. **12**

Table des notations. **13**

1 Introduction. **15**
1.1 Contexte de l'étude. 15
1.2 Les matériaux cimentaires. 18
1.3 La théorie de l'homogénéisation. 19
1.4 Déterminer la diffusivité équivalente. 21
1.5 Les méthodes de simulation multi-échelles. 22
1.6 Plan du manuscrit. 23

I Aspects théoriques. **27**

2 État de l'art. **29**
2.1 Présentation des méthodes multi-échelles. 30
2.2 Méthodes de Galerkin directes. 32
2.2.1 Présentation de la méthode. 32
2.2.2 Améliorations. 33
2.2.3 Extensions et applications. 34
2.3 Méthodes de Galerkin approchées. 35
2.3.1 Présentation de la méthode. 35
2.3.2 Extensions et applications. 38
2.4 Méthodes des Volumes Finis multi-échelles. 39
2.4.1 Présentation de la méthode. 39
2.4.2 Extensions et applications. 41

3 Méthodes multi-échelles pour les matériaux cimentaires. **43**
3.1 Découpage du domaine Ω. 45
3.2 Résolution des problèmes de cellules. 46
3.2.1 Problèmes de cellules et Volumes Finis. 46
3.2.2 La méthode de Volumes Finis à neuf points *VF9*. 48
3.2.3 La méthode de Volumes Finis *VFDiam*. 50

3.3 Résolution du problème grossier. 53
3.3.1 Méthode des Éléments Finis Q_1. 54
3.3.2 Méthode discontinue de Galerkin. 58
3.4 Reconstruction de la solution et du flux au niveau fin. 64
3.5 Estimation du coût de calcul. 64
3.6 Modélisation des matériaux cimentaires. 66
3.6.1 Description générale. 66
3.6.2 Dégradation des matériaux cimentaires. 69

II Mise en œuvre en dimension 2. 71

4 Implémentation de la méthode Q_1/*VF9* en dimension deux. 73
4.1 Implémentation. 74
4.1.1 Qualification du module FCTVF9. 76
4.1.2 Qualification du module CREMK. 78
4.1.3 Qualification du module CREASSEMBLAGE. 81
4.2 Validation de la méthode Q_1/*VF9* sur un exemple théorique. . . . 83
4.2.1 Convergence théorique. 83
4.2.2 Validation. 85
4.3 Application aux matériaux cimentaire. 89
4.3.1 Génération d'un matériau cimentaire *via* le module COMBS. 89
4.3.2 Discrétisation du milieu. 90
4.3.3 Un exemple de pâte cimentaire. 93
4.4 Conclusions. 101

III Mise en œuvre en dimension 3. 103

5 Mise en œuvre dans le cas 3*D* : simulations *MPCube*. 105
5.1 La chaîne de calcul multi-échelle *SALOME-MPCube*. 106
5.1.1 Génération des maillages. 106
5.1.2 Préparation des calculs. 108
5.1.3 Résolution des problèmes de cellules. 109
5.1.4 Construction de la base locale et calcul des matrices locales. 110
5.1.5 Mise en forme des résultats locaux. 112
5.1.6 Assemblage et résolution du problème grossier. 114
5.1.7 Reconstruction de la solution fine. 115
5.2 Implémentation et qualification. 116
5.2.1 Le code de calcul multi-physique *MPCube*. 117
5.2.2 Construction des fonctions locales de Galerkin. 117
5.2.3 Résolution du problème grossier. 125

6 Mise en œuvre dans le cas 3*D* : générations et opérations sur les maillages. 131
6.1 La plate-forme *SALOME*. 132
6.1.1 Description des besoins d'une chaîne de calcul multi-échelle. 132
6.1.2 Présentation de la plate-forme *SALOME*. 134
6.1.3 Composants utilisés en amont des simulations numériques. 135
6.1.4 Composants utilisés en aval des simulations numériques. . 136
6.2 Génération automatique des maillages de cellules. 136
6.2.1 Notations. 136
6.2.2 Principes généraux. 138
6.2.3 Adaptation de la fonction native MAKEPARTITION. 143
6.2.4 Gestion des points de tangence. 152
6.2.5 Test de l'algorithme complet. 154
6.3 Génération de maillages coïncidents. 155
6.3.1 Problématiques et idées de solutions. 155
6.3.2 Description algorithmique. 160
6.4 Évolution des maillages avec la dégradation du milieu. 162
6.4.1 Présentation de la méthode. 162
6.4.2 Description algorithmique. 164

7 Mise en œuvre dans le cas 3*D* : simulations. 169
7.1 Exemples théoriques. 170
7.1.1 Qualification de la chaîne de calcul *SALOME-MPCube*. . 170
7.1.2 Cas d'un milieu périodique. 173
7.1.3 Réponse de la chaîne de calcul à un grand nombre de macroéléments. 177
7.2 Applications aux matériaux cimentaires. 182
7.2.1 Application à un échantillon de mortier. 182
7.2.2 Application à un échantillon périodique de pâte de ciment. 188
7.2.3 Application à un échantillon aléatoire de pâte de ciment. . 194

8 Conclusions et perspectives. 201
8.1 La chaîne multi-échelle *SALOME-MPCube*. 201
8.1.1 Parallélisme. 202
8.1.2 Génération automatique des maillages. 203
8.2 Homogénéisation des matériaux cimentaires. 205
8.2.1 Homogénéisation numérique. 205
8.2.2 Modélisation des matériaux cimentaires. 205
8.2.3 Dégradation des matériaux cimentaires. 207
8.3 Aspects théoriques. 208
8.3.1 Les méthodes multi-échelles. 208
8.3.2 Extensions du problème. 209

IV Annexes. **211**

A Schéma *VFDiam* en dimension 3. **213**

B Codes sources et jeux de données de la chaîne de calcul *SALOME-MPCube*. **217**

B.1 Génération et opérations sur les maillages. 217

B.1.1 La méthode MAKEINTERSECTIONBYSHELLS. 217

B.1.2 La méthode MAKECUTBYSHELLS. 218

B.1.3 La méthode MAKECONFORMTOTANGENTOBJECTS. . . 219

B.2 Exemples de jeux de données *MPCube*. 221

B.2.1 Résolution d'un problème. 221

B.2.2 Renumérotation d'un maillage. 223

C Publications. **225**

Table des figures. **239**

Table des algorithmes. **243**

Bibliographie. **245**

Principales notations.

Problème de diffusion.

d	dimension de l'espace de travail
Ω	domaine de travail
Γ	frontière du domaine Ω
D	diffusivité du domaine
C	solution du problème
f	terme source
g_D, Γ_D	fonction et lieu des conditions aux limites de Dirichlet
g_N, Γ_N	fonction et lieu des conditions aux limites de Neumann
$\mathbf{n}$	vecteur normal unitaire extérieur à Ω.

Méthode multi-échelle.

$\mathscr{D}$	division du milieu
ρ	taux de sur-échantillonnage
$(\mathscr{D}, \rho)$	découpage du milieu
h	taille caractéristique de l'échelle fine
H	taille caractéristique de l'échelle grossière
α	rapport h sur H
$C_{H,h}$	Solution fine reconstruite

Problème fin.

K	macroélément
$\hat{K}$	cellule construite à partir de K
n	nombre de problèmes de cellules de K
$(S_l)_{1\leq l\leq n}$	sommets du macroélément
$(\hat{S}_l)_{1\leq l\leq n}$	sommets de la cellule
$T_h(\hat{K})$	maillage fin de $\hat{K}$
k	élément de $T_h(\hat{K})$
f	face de $T_h(\hat{K})$
$(\Psi^l_{\hat{K}})_{1\leq l\leq n}$	solutions des problème de cellule
$(\Phi^l_K)_{1\leq l\leq n}$	base locale de K

Problème grossier.

$T_H(\Omega)$	maillage grossier du domaine Ω.
K	élément de $T_H(\Omega)$ ou macroélément
F	face de $T_H(\Omega)$
V_H	espace de Galerkin
$\mathscr{I}$	ensemble des degrés de liberté de V_H
$(\Phi^I)_{I\in\mathscr{I}}$	base de l'espace V_H
C_H	solution grossière
$\mathbb{K}$, $\mathbb{M}$, $\mathbb{M}_b$	matrices de raideur, de masse et de masse de bord
$\mathbb{T}$, $\mathbb{Y}$	matrice de saut, de pénalité
μ	terme de pénalité

Géométrie du domaine.

$\mathscr{I}$	ensemble des inclusions du domaine Ω
$\mathscr{I}^K$	ensemble des inclusions de la cellule $\hat{K}$
K_0	macroélément homogène
$\hat{K}_0$	cellule homogène
K_0^c	couronne homogène

Autres.

∂A	frontière du domaine A
$\mathscr{O}()$	notation de Landau
δ	symbole de Kronecker
D^*	diffusivité équivalente

Chapitre 1

Introduction.

1.1 Contexte de l'étude.

La loi du 30 décembre 1991 sur la gestion des déchets radioactifs de haute activité et à vie longue a confié à l'Andra, l'Agence nationale pour la gestion des déchets radioactifs, la tâche d'évaluer la possibilité d'un stockage réversible des déchets en formation géologique profonde, notamment par la construction d'un laboratoire souterrain. Cette même loi confie deux autres axes de recherche au Commissariat à l'Énergie Atomique ; devenu depuis le Commissariat à l'Énergie Atomique et aux Énergies Alternatives (CEA) : d'une part la séparation des éléments à vie longue, associée à la réduction de la durée de vie des plus toxiques d'entre eux (transmutation), et d'autre part le conditionnement et l'entreposage de longue durée en surface ou en faible profondeur. Sont également associés à cette mission l'Autorité de sûreté nucléaire et son appui technique, l'Institut de Radioprotection et de Sûreté Nucléaire (IRSN), afin d'examiner les résultats de recherche sous l'angle de la sûreté.

En 2005, un rapport global d'évaluation des recherches a été publié à l'intention du Parlement, afin de nourrir le débat parlementaire de 2006 [24]. Ce dernier, par la loi-programme du 28 juin 2006 sur la gestion durable des matières et déchets radioactifs, a élargi le rôle de l'Andra. Sa mission est désormais de trouver, mettre en œuvre et garantir des solutions de gestion sûres, sur le long terme, pour tous les déchets radioactifs français. Dans ce cadre, elle a mis en place, en collaboration avec le CEA et l'IRSN, d'importants programmes de Recherche et Développement concentrés sur deux points. Il s'agit d'une part de concevoir les technologies nécessaires à l'entreposage temporaire et au stockage réversible des déchets et d'autre part d'effectuer des études scientifiques précises pour garantir la pérennité, à très long terme, des solutions de stockage et de la protection que celles-ci apportent à l'homme et à l'environnement. C'est sur ce second point que le Laboratoire de Simulation des Écoulements et Transports (LSET) a été saisi, notamment dans le cadre de la réalisation de la plate-forme de simulation *Alliances* [23, 44].

Le LSET développe également des solutions de simulations (méthodes numériques, applications) dédiées aux matériaux cimentaires comme le béton. Les bétons et ciments interviennent en effet à tous les niveaux du processus de stockage, à commencer par la réalisation des structures du futur centre de stockage en milieu géologique. Se pose alors la question de la durabilité de ces ouvrages, un problème récurrent dans le domaine du génie civil. La question est d'autant plus importante que les durées relatives à la gestion des déchets sont longues.

Le béton, de part ses propriétés de confinement, joue également un rôle important dans le traitement des déchets de moyenne activité à vie longue, dit *déchets B*. Il s'agit principalement de déchets technologiques issus du processus de traitement des combustibles usés ou d'activité de recherche. En 2005, les déchets B représentaient 4.6% du volume des déchets nucléaires français, pour environ 8% de la radioactivité totale. Ils représentent donc une composante charnière des déchets radioactifs, à mi-chemin entre les *déchets A* de faible activité (95% du volume pour 0.068% de la radioactivité totale), et les *déchets C* de haute activité (0.2% du volume pour 91.7% de la radioactivité totale). Cette position de transition explique pourquoi les colis de déchets B prennent une grande variété de forme : poudre métallique, bloc de bitume ou, dans la majorité des cas, de béton. Les solutions de stockage des déchets B prévoient en outre d'enchasser plusieurs colis de déchets dans un surconteneur, lui aussi en béton [25].

On suppose généralement que ces surconteneurs sont perméables à l'eau, et qu'ils subissent de cette façon un important processus de corrosion. De plus, en présence d'eau, la radioactivité contenue dans les colis est relachée, même si le flux de radionucléides est limité, au sein du béton, par des phénomènes de précipitation et de sorption. La question est donc de prédire au mieux, à partir d'expériences pratiques de corrosion et de simulations numériques, d'une part le devenir du béton au cours du temps ; ainsi que l'évolution associée des propriétés mécaniques et de confinement ; d'autre part le transport des radionucléides dans le béton en fonction de son niveau de dégradation.

En partie mené par le LSET, le projet IOLS, Infrastructure et Outils Logiciel pour le Simulation [45], a ainsi permis de développer la chaîne de calcul *ChaCalBe* pour la prédiction de l'évolution des propriétés effectives des bétons (performances, durabilité), plus précisément pour la description de la dégradation physique et chimique à long terme en milieu saturé des matériaux cimentaires. La finalité de cet outil est de décrire et prédire numériquement l'évolution des matériaux, c'est-à-dire d'explorer le lien entre la corrosion et l'évolution des propriétés mécaniques et diffusives des bétons. Il est à noter que la chaîne de calcul *ChaCalBe* fait intervenir le code de calcul *MPCube* [61], un outil également utilisé au cours de cette thèse.

Une partie des travaux de cette thèse s'inscrit dans le cadre du projet EHPOC (Environnement Haute Performance pour l'Optimisation et la Conception [93]) qui a fait suite au projet IOLS. Il s'agit d'une mise en œuvre de la chaîne de calcul *ChaCalBe* sur des volumes représentatifs de matériaux cimentaires, en réalisant des

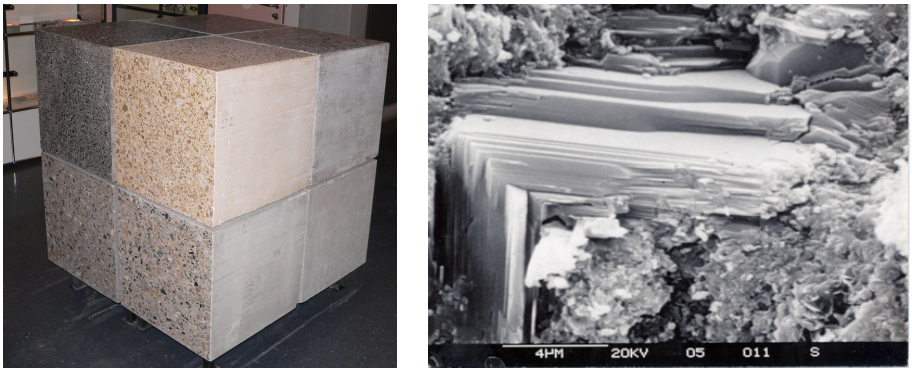


FIGURE 1.1 – La structure apparente du béton change en fonction de l'échelle d'observation. À gauche, des blocs de béton de 50 cm de côté, mélange solide de ciment et de graviers (Image Wikimedia Commons). À droite, une vue microscopique d'un ciment Portland hydraté : cristaux de portlandite $Ca(OH)_2$ dans une matrice de silicates de calcium hydratés nanocristallisés $C-S-H$ (Image CEA).

calculs relatifs à la dégradation chimique de ceux-ci. Ils ont d'ailleurs fait l'objet d'un rapport interne CEA par *Chomat* et al. [70].

Les travaux de cette thèse se sont concentrés sur le phénomène de diffusion. Soit Ω le domaine d'étude, le transport d'un élément chimique par diffusion au sein d'un milieu poreux comme le béton est modélisé, en l'absence de réactions chimiques, par le problème aux limites suivant [81] :

$$\begin{cases} -\nabla\cdot(D\nabla C) &= f & \text{dans } \Omega, \\ C &= g_D & \text{sur } \Gamma_D, \\ D\nabla C\cdot\mathbf{n} &= g_N & \text{sur } \Gamma_N, \end{cases} \tag{1.1}$$

où (Γ_D,Γ_N) est une partition de $\Gamma=\partial\Omega$ la frontière du domaine Ω. D est la *diffusivité* du milieu. Il s'agit d'une matrice symétrique définie positive de taille $d\times d$, où d est la dimension du problème ($d=2$ ou $d=3$). f est un terme source, g_D et g_N sont respectivement la condition aux limites de Dirichlet et de Neumann.

L'équation (1.1) en elle-même est un problème modèle bien connu en analyse numérique [50, 71, 94, 104, 157]. La pierre d'achoppement tient ici dans l'application de cette équation aux matériaux cimentaires, qui possèdent une structure physique très complexe. On doit ici résoudre un problème à plusieurs échelles en ce sens que, pour étudier des ouvrages en béton d'un ou plusieurs mètres de long, il est nécessaire de prendre en compte des structures minérales de quelques micromètres de diamètre. La Figure 1.1 illustre ces deux échelles de travail.

Dans ce contexte, les méthodes numériques normalement utilisées pour obtenir une approximation de la solution de (1.1) sont inadaptées, voire tout simplement

inutilisables. En effet, elles requerraient de discrétiser très finement le domaine de travail, au moins jusqu'à la taille des microstructures du ciment, et si possible en dessous. Les maillages ainsi obtenus atteindraient alors des tailles défiant toute résolution numérique, même sur les plus récents supercalculateurs. Il existe donc bel et bien un besoin important de méthodes numériques multi-échelles dédiées aux matériaux cimentaires, et c'est dans ce cadre que s'inscrivent ces travaux de thèse.

1.2 Les matériaux cimentaires.

Le *béton* est un matériau de construction constitué de granulats, généralement du sable et des gravillons, agglomérés en un tout unique par un liant, le *ciment*. La poudre de ciment est un mélange complexe de silice, d'alumine, de carbonate de chaux que l'on a traité chimiquement et mécaniquement. C'est le mélange de l'eau, de la poudre de ciment et des granulats qui forme, une fois séché, le béton. On parle de *mortier* quand les granulats utilisés sont très fins, se réduisant généralement à du sable. Dans le présent ouvrage, les *matériaux cimentaires* regroupent les pâtes de ciment, c'est-à-dire un ciment hydraté puis séché, mais aussi tous les bétons et mortiers que l'on a pu façonner avec le ciment.

Le ciment est un matériau très ancien et de nombreuses preuves de son utilisation au cours de l'Histoire Antique sont avérées. Les Egyptiens utilisaient déjà un mélange de chaux, d'argile, de sable et d'eau près de 2600 ans avant notre ère, pour la construction de leurs pyramides [59]. Les Romains employaient l'*opus caementicium* de manière quasi-systématique dans leurs constructions, ce qui explique la longevité de nombre de leurs œuvres [22]. On considère généralement que l'usage du béton s'est perdu pendant la période du Moyen-Âge, avant d'être redécouvert au XV^{e} siècle. La reconstruction, par Fra Giovanni Giocondo, du quai d'escale du Pont de Notre Dame à Paris est la première utilisation reconnue du béton à l'époque moderne [100].

Il faudra cependant attendre la seconde moitié du XVIII^{e} pour que des recherches scientifiques précises sur le ciment [66, 117] permettent l'essor industriel des ciments, et par là-même, une généralisation de l'utilisation du béton dans la construction. En France, la découverte du ciment est attribuée à l'ingénieur des Ponts et Chaussées Louis Vicat, qui fut le premier à fabriquer de manière artificielle et contrôlée, des chaux hydrauliques dont il détermina les composants et leurs proportions [172]. C'est cependant l'industriel Joseph Aspdin qui déposa en 1824 le premier brevet qui crée la marque *ciment de Portland*. Les premières usines françaises de ciment Portland artificiel datent de 1850, celles de ciment de grappier de 1870. La fabrication de ciment de laitier date de 1890. On peut également citer, en 1908, la découverte du *Ciment Fondu*© par Jules Bied, directeur du laboratoire de recherche de la société Pavin de Lafarge, un ciment à base de calcaire et de bauxite, qui résiste aux agressions et aux hautes températures.

De part leurs constructions, les matériaux cimentaires, ciments et mortiers, sont le siège de très grandes hétérogénéités. Leurs propriétés physiques, comme la diffusivité, varient fortement sur le domaine d'étude, non seulement d'un point à l'autre, mais aussi en fonction de l'échelle à laquelle on observe le milieu. Par exemple, à l'échelle du béton, où un *volume élémentaire représentatif* (VER) est de l'ordre du centimètre, la pâte cimentaire est considérée comme homogène et enchasse des agrégats (graviers, sable, poussières). Si on se place à l'échelle microstructurale de la matrice cimentaire (VER inférieur à $100\mu m$), on est obligé de prendre en compte des microstructures minérales dont la taille varie entre 1 et $30\mu m$ environ. C'est pourquoi on parle de problèmes multi-échelles.

1.3 La théorie de l'homogénéisation.

Dans le cadre d'un problème multi-échelle, la description du milieu dépend de l'échelle à laquelle on se place. Plutôt qu'un unique problème à résoudre, on dispose en fait d'une famille de problèmes dépendant d'un paramètre ε caractérisant l'échelle à laquelle on se trouve. Par exemple, dans le cadre du béton, on peut choisir une taille, en-deça de laquelle les structures minérales sont ignorées, et considérées comme faisant partie du fond homogène. Pour chaque $\varepsilon > 0$, on peut donc résoudre un problème du type suivant :

$$\left\{\begin{array}{rcll} -\nabla\cdot(D^\varepsilon\,\nabla C^\varepsilon) &=& f & \text{dans } \Omega,\\ C^\varepsilon &=& g_D & \text{sur } \Gamma_D,\\ D^\varepsilon\,\nabla C^\varepsilon\cdot\mathbf{n} &=& g_N & \text{sur } \Gamma_N, \end{array}\right. \tag{1.2}$$

où Γ_D, Γ_N, f, g_D et g_N ont été définis plus haut.

L'étude mathématique de la limite des problèmes (1.2) et de la suite $(C^\varepsilon)_{\varepsilon>0}$ quand ε tend vers 0 relève de *la théorie de l'homogénéisation*. Il s'agit d'une stratégie courante pour aborder les problèmes multi-échelles, qui est abondamment décrite dans la littérature scientifique [26, 27, 34, 46, 72, 132, 145, 164, 170].

On ne citera ici que le résultat principal de la théorie qui stipule que, quand ε tend vers 0, sous certaines conditions sur D^ε et à l'extraction d'une sous-suite de $(C^\varepsilon)_{\varepsilon>0}$ près, les suites $(C^\varepsilon)_{\varepsilon>0}$ et $(D^\varepsilon\nabla C^\varepsilon)_{\varepsilon>0}$ convergent. De plus, il existe $D^\star$ tel que :

$$C^\varepsilon \rightharpoonup C^\star, \tag{1.3}$$

$$D^\varepsilon\nabla C^\varepsilon \rightharpoonup D^\star\nabla C^\star. \tag{1.4}$$

En outre $C^\star$ est la solution du *problème homogénéisé* :

$$\left\{\begin{array}{rcll} -\nabla\cdot(D^\star\,\nabla C^\star) &=& f & \text{dans } \Omega,\\ C^\star &=& g_D & \text{sur } \Gamma_D,\\ D^\star\,\nabla C^\star\cdot\mathbf{n} &=& g_N & \text{sur } \Gamma_N, \end{array}\right. \tag{1.5}$$

Le point important de ce théorème est que $D^\star$ ne dépend ni du terme source f ni des conditions aux limites g_D et g_N du problème. Le terme $D^\star$ ne dépend que de la suite $(D^\varepsilon)_{\varepsilon>0}$.

Ces limites sont à comprendre au sens faible [54] ou, en d'autres termes, au sens moyenné. Cela signifie que l'on a, pour des fonctions tests f scalaires et $\mathbf{g}$ vectorielles :

$$\int_\Omega C^\varepsilon f \to \int_\Omega C^\star f$$
$$\int_\Omega D^\varepsilon \nabla C^\varepsilon \cdot \mathbf{g} \to \int_\Omega D^\star \nabla C^\star \cdot \mathbf{g}$$

On se reportera aux ouvrages précédemment cités pour une description plus précise de ces convergences et des espaces fonctionnels impliqués.

En un sens physique, $D^\star$ représente la diffusivité d'un matériau équivalent, dit *homogénéisé*, qui ne dépend que de la description faite du milieu à chaque échelle, indépendamment des transformations qu'on lui impose. Disposer de la version homogénéisée du domaine, c'est donc connaître la réponse *globale*, *macroscopique* du milieu aux actions extérieures : une réponse qu'il est généralement facile de déterminer, théoriquement ou numériquement, puisque $D^\star$ varie seulement à une échelle macroscopique. Les réponses *locales*, c'est-à-dire dépendant de l'échelle fine ε, peuvent être calculées par le biais d'un développement limité :

$$C^\varepsilon \approx C^\star + C_1^\varepsilon \tag{1.6}$$

où C_1^ε est la solution d'un problème dérivé de (1.2), dit de *cellule* (cf (1.9) dans le cas périodique). C_1^ε, qu'on appelle généralement *terme correcteur*, tend vers 0 avec ε mais son gradient ∇C_1^ε contribue fortement aux variations de C^ε quel que soit ε. Le terme correcteur joue donc un rôle fondamental dans l'estimation des fluctuations locales de la solution C^ε. On trouvera plus de détails sur ce point dans [132, 145].

Dans le cadre *périodique*, c'est-à-dire où $D^\varepsilon = d(\frac{x}{\varepsilon})$ avec d une matrice périodique de cellule de périodicité Y, il est possible d'itérer le processus et de réaliser des développements limités de C^ε à des ordres supérieurs en ε [53]. On peut alors expliciter la dépendance en ε des différents termes sous la forme suivante :

$$C^\varepsilon \approx C^\star + \varepsilon C_1 + \varepsilon^2 C_2 \tag{1.7}$$

Malheureusement, si on sait caractériser $D^\star$, il n'est possible de connaitre explicitement cette valeur que dans certains cas très particuliers. Par exemple, si on se place dans le cadre périodique on a :

$$D^\star_{i,j} = \frac{1}{|Y|}\int_Y d(e_i + \nabla w_i)(e_j + \nabla w_j) \tag{1.8}$$

où $(e_i)_{1\leq i\leq d}$ sont les vecteurs cardinaux de $\mathbb{R}^d$, et $(w_i)_{1\leq i\leq d}$ les solutions des problèmes, dits *de cellule* :

$$\left\{\begin{array}{rcll} -\nabla_y\cdot\left(d\left(e_i+\nabla_y w_i\right)\right) & = & 0 & \text{sur } Y. \\ w_i \text{ périodique} & & & \text{sur } \partial Y. \end{array}\right. \tag{1.9}$$

En dimension 2, il est par exemple possible d'obtenir explicitement le coefficient homogénéisé correspondant à un matériau laminé ou en échiquier [132]. Certains exemples non périodiques existent aussi : on peut décrire explicitement l'homogénéisation d'un milieu empli de sphères concentriques de différentes tailles [143]. Il s'agit cependant de cas particuliers et, en général, on ne peut accèder aux valeurs de $D^\star$ que par le calcul numérique [106, 107, 108].

1.4 Déterminer la diffusivité équivalente.

Dans le cas pratique étudié ici, D^ε n'est pas une fonction périodique, et on ne peut déterminer explicitement $D^\star$ ni même la caractériser par une formule similaire à (1.8). A fortiori, on ne peut espérer déterminer explicitement $C^\star$ pour un milieu comme le béton. Du reste, on ne dispose pas de la suite complète $(D^\varepsilon)_{\varepsilon>0}$, mais seulement de la description du milieu en un nombre limité d'observations, généralement parce qu'il n'est pas possible d'accéder aux informations plus fines (limites des instruments de mesures).

Quitte à s'écarter du cadre mathématique strict de l'homogénéisation, on va donc plutôt chercher à déterminer la diffusivité *équivalente* D^* de D^ε, c'est-à-dire une diffusivité qui rendrait compte, à l'échelle macroscopique, des structures fines présentes à l'échelle d'observation. Dans le cas des matériaux cimentaires, il s'agit par exemple de déterminer la diffusivité de la pâte de ciment (que l'on utilisera lors de simulations sur le béton, à l'échelle du centimètre) à partir de sa description fine de celle-ci (à l'échelle du micromètre).

La diffusivité équivalente est liée à la définition d'un *Volume Élémentaire Représentatif* (VER). Il s'agit d'un échantillon type du matériau, dont on suppose qu'il renferme toutes les structures, toutes les informations nécessaires pour prédire correctement le comportement macroscopique du matériau. À l'échelle macroscopique, un volume quelconque du matériau pourra donc être remplacé par un assemblage de ses VER sans affecter les résultats des simulations.

La diffusivité équivalente D^* doit être cohérente avec la définition du VER du milieu et avec le comportement attendu à l'échelle supérieure. Par exemple, la pâte de ciment se comporte comme un matériau homogène isotrope à l'échelle du béton et la valeur de D^* calculée à partir du VER doit donc l'être aussi.

On peut calculer D^* de plusieurs façons. Une première méthode est de résoudre (1.9) sur le domaine Ω complet, qui serait donc assimilé à la cellule de périodicité Y de notre diffusivité. D^* serait ensuite calculée *via* l'équation (1.8) avec $Y=\Omega$.

On peut également résoudre numériquement un problème copié sur les mesures expérimentales de diffusivité équivalente [48, 49, 63, 73] :

$$\begin{cases} -\nabla \cdot (D\nabla C) &= 0 \quad \text{sur } \Omega = [0, L_i]^d \\ C(0,y,z) &= 1 \\ C(L_x,y,z) &= 0 \\ D\nabla C \cdot n &= 0 \quad \text{sur les autres bords,} \end{cases} \tag{1.10}$$

On calcule ensuite D_x^*, une valeur équivalente selon la direction x, à partir des flux entrant et sortant :

$$D_x^* = \frac{1}{L_x} \int_{\{x=L_x\}} D\nabla C \cdot n$$

En répétant le processus pour chaque dimension du domaine, on obtiendra une matrice équivalente D^* diagonale.

Ce problème a déjà été abordé par les laboratoires du CEA sous les angles de l'analyse physique et de la modélisation mathématique. La piste de l'homogénéisation analytique [41, 42, 84, 135] peut s'avérer difficile à mettre en œuvre, notamment à cause de la variation continue et de la distribution spatiale aléatoire des inclusions (agrégats ou phase minérale selon l'échelle) dans le milieu. Reste donc la modélisation directe en trois dimensions [21, 36, 43, 70] qui se heurte malheureusement à l'explosion des temps de calculs. Afin de poursuivre plus en avant cette approche du problème, il est donc nécessaire de développer des méthodes spécifiques de résolution numérique multi-échelles, et c'est dans ce sens que les travaux de cette thèse ont été effectués.

1.5 Les méthodes de simulation multi-échelles.

On considère le problème (1.2), défini sur le domaine Ω et dépendant du paramètre ε. Pour le résoudre par une méthode de simulation numérique, par exemple les Éléments Finis ou les Volumes Finis, il est nécessaire de discrétiser le domaine Ω afin d'obtenir un maillage $T_h(\Omega)$ de pas h. Comme ε est la longueur caractéristique du milieu, il faut choisir $h < \varepsilon$, sous peine d'ignorer entièrement la description fine du milieu. En conséquence de quoi, le nombre d'éléments composant $T_h(\hat{K})$ augmente très rapidement quand ε diminue, jusqu'à dépasser les capacités de résolution actuellement disponibles.

Plusieurs axes de recherches ont été étudiés afin de s'affranchir de ce problème. Ainsi, les méthodes de décomposition de domaine résolvent le problème portion du domaine par portion du domaine, itérant le processus jusqu'à obtenir une solution globale correcte [133]. Une autre approche, dans laquelle s'inscrivent ces travaux de thèse, a été de développer des méthodes de simulation multi-échelles. Le principe est de résoudre le problème (1.2) sur un maillage *grossier* $T_H(\Omega)$, avec $H > \varepsilon$, mais de calculer tous les termes de la méthode numérique choisie (fonctions de

bases et matrices des Éléments Finis, flux des Volumes Finis) sur une portion d'un maillage *fin* $T_h(\Omega)$ du domaine. De cette manière, on intègre au problème grossier les informations *locales*, c'est-à-dire issues de l'échelle fine du problème. À partir de la solution du problème grossier et des termes fins, par exemples les fonctions de bases, on parvient ensuite à reconstruire une solution fine sur tout ou partie du maillage $T_h(\Omega)$.

Historiquement, la méthode des Éléments Finis fut la première à être adaptée au contexte multi-échelle [28, 86, 120, 121, 140, 141]. Dans ce cas, le maillage fin permet de déterminer une base d'Éléments Finis adaptée à notre milieu. On doit pour cela résoudre plusieurs problèmes *locaux* sur des portions du domaine. Ces problèmes sont généralement résolus par une méthode d'Éléments Finis.

On a vu depuis apparaître des méthodes multi-échelles centrées sur les Éléments Finis non-conformes [69, 90], les Volumes Finis [111, 125, 127] ou encore les méthodes de Galerkin Discontinues [3, 16, 58, 122, 124]. Ces méthodes multi-échelles feront l'objet d'une étude plus détaillée au chapitre §2. La justification théorique et les estimations d'erreurs des méthodes de simulation multi-échelles se font généralement dans le cadre de l'homogénéisation périodique.

Les applications des méthodes de simulation multi-échelles ont surtout porté sur les problématiques de l'industrie pétrolière (transport eau-huile en milieu poreux) [1, 4, 120, 125, 126] et ceux du stockage géologique du dioxyde de carbone [112, 138, 177] mais très peu, du moins à notre connaissance, sur les matériaux cimentaires [88].

On a développé au cours de ces travaux de thèse de nouvelles méthodes multi-échelles couplant, à l'échelle grossière, une méthode d'Éléments Finis ou de Galerkin discontinue avec des méthodes de Volumes Finis à l'échelle fine. Ces méthodes s'appuient sur les précédents travaux sur le sujet dans le cadre Éléments Finis [28, 120], mais utilisent des méthodes de Volumes Finis à l'échelle fine au lieu des méthodes d'Éléments Finis. On compte ainsi augmenter la stabilité des méthodes multi-échelles face aux fortes discontinuités et à l'anisotropie des matériaux cimentaires, d'autant plus que les méthodes de Volumes Finis utilisées *VF9* [97] et *VFDiam* [20, 79] ont été développées dans ce but. Le couplage de ces méthodes est détaillé au chapitre §3.

1.6 Plan du manuscrit.

Cette introduction terminée, le lecteur trouvera dans le chapitre suivant un état de l'art dédié aux méthodes multi-échelles (§2). Le chapitre suivant est consacré aux nouvelles méthodes multi-échelles développées au cours de ces travaux de thèse (§3). On y présente en détail les caractéristiques originales de ces méthodes, notamment le couplage entre les méthodes de Volumes Finis à l'échelle fine et celles d'Éléments Finis à l'échelle grossière.

La suite du manuscrit s'articule autour des deux implémentations réalisées au cours de la thèse. En effet, la première méthode développée, que l'on appelle Q_1/*VF9*, a fait l'objet d'une implémentation indépendante en Python, dans un cadre $2D$. Cette implémentation, et les résultats que l'on a obtenus grâce à elle, sont présentés au chapitre §4. Cette maquette avait pour but principal de prendre rapidement conscience des diverses problématiques du sujet de thèse. C'est pourquoi on s'est affranchi autant que possible des contraintes techniques, en imposant des hypothèses fortes sur le problème, comme d'utiliser uniquement des maillages quadrangulaires réguliers.

On a ainsi appliqué la méthode Q_1/*VF9* à plusieurs cas théoriques et à celui, plus complexe, d'une pâte cimentaire $2D$. Plusieurs problèmes ont ainsi été mis à jour. En effet, si la méthode converge bien sur les cas théoriques, en pratique l'erreur a tendance à stagner pour le cas de la pâte cimentaire. La responsabilité en revient aux forts sauts de diffusivité du milieu, sauts dont l'ordre de grandeur peut atteindre 10^8. Il semble difficile, voir impossible, de choisir optimalement les différents paramètres de la méthode dans un cas aussi raide. En conséquence de quoi, la méthode de recouvrement [79], qui permet normalement de corriger certains défauts de la méthode, est ici inefficace.

On a proposé plusieurs modifications afin d'améliorer les résultats de la méthode. Tout d'abord, la taille des sytèmes étudiés a été augmentée. Augmenter la précision de la discrétisation permet en effet une capture plus fine des effets de couche limites *via* la méthode de recouvrements. Comme les calculs présentés au chapitre §4 atteignent les capacités maximales d'un ordinateur séquentiel, il a été nécessaire de changer le cadre d'implémentation de cette méthode (programme Python sur un ordinateur) pour une interface plus robuste. Le choix s'est porté sur le code *MPCube* [61], basé sur le noyau de calcul *Trio-U* [171].

Le cadre de travail a été considérablement élargi, l'ambition affichée étant de pouvoir traiter des matériaux cimentaires $2D$ et $3D$. Le chapitre §5 présente ainsi les deux méthodes implémentées au sein du code *MPCube*, la première issue des Éléments Finis Q_1/*VFDiam*, la seconde apparentée aux méthodes de Galerkin discontinues GD/*VFDiam*.

Afin de mieux modéliser les géométries des matériaux cimentaires, la contrainte sur les maillages a été supprimée. Les maillages seront maintenant quelconques et devront être générés avant toute simulation. Le chapitre §6 décrit l'important travail réalisé au sein de la plate-forme *SALOME* [163] afin de pouvoir discrétiser rapidement et automatiquement les domaines de travail. Il décrit ainsi comment l'on obtient, à partir d'une description du matériau cimentaire, les maillages nécessaires aux simulations *MPCube*.

On présente au chapitre §7 les simulations effectuées par le biais de la chaîne de calcul *SALOME-MPCube*. Certaines simulations sont théoriques (§7.1), tandis que d'autres concernent des exemples de matériaux cimentaires (§7.2). En particulier, la section §7.2.3 présente un exemple représentatif des différents ordres de grandeur

que l'on rencontre en travaillant sur des matériaux cimentaires. Le milieu présente en effet de multiples sauts de diffusivité d'un facteur 1×10^{11}, conséquences de l'existence de plusieurs dizaines de milliers de microstructures. La discrétisation complète de ce milieu demande près de 100 millions de tétrahèdres, ce qui exclut toute résolution d'un problème de diffusion par une méthode numérique classique.

Enfin, le chapitre §8 fait le bilan des capacités de la chaîne de calcul multi-échelle *SALOME-MPCube*. On y reprend notamment les remarques ponctuelles que l'on a faites au cours des trois précédents chapitres sur les forces et faiblesses des méthodes développées pendant ces travaux de thèse. Des pistes de travails sont proposées pour améliorer la chaîne de calcul et pour en étendre les applications.

Les résultats présentés dans ce manuscrit de thèse ont été partiellement rédigés dans deux rapports internes CEA [9, 70] et un article [14], reproduit à l'Annexe §C. Ils ont également fait l'objet de communications orales [10, 12, 13] et écrites [7, 8, 11] lors de conférences nationales et internationales.

Première partie

Aspects théoriques.

Chapitre 2

État de l'art.

Introduction.

Ce chapitre présente une synthèse des différentes méthodes multi-échelles que l'on peut trouver dans la littérature scientifique, afin de résoudre numériquement le problème suivant, ou une de ses généralisations :

$$\left\{\begin{array}{rcll} -\nabla\cdot(D\nabla C) & = & f & \text{dans } \Omega, \\ C & = & g_D & \text{sur } \Gamma_D, \\ D\nabla C\cdot\mathbf{n} & = & g_N & \text{sur } \Gamma_N, \end{array}\right. \tag{2.1}$$

où (Γ_D,Γ_N) forme une partition de $\Gamma=\partial\Omega$ la frontière du domaine Ω. f, g_D et g_N sont respectivement le terme source, la condition aux limites de Dirichlet et la condition aux limites de Neumann du problème.

On suppose que ce problème comporte deux échelles. Il y a d'une part l'échelle *globale* qui correspond à la taille du domaine Ω, et d'autre part il y a l'échelle *fine*, qui correspond à la taille caractéristique des fluctuations du milieu. On suppose que la taille de ces fluctuations, intégrées au problème par le biais de la matrice D, est très petite par rapport à la taille du domaine. Il y a donc un véritable écart entre les deux échelles.

L'échelle fine représente bien entendu une information plus précise que l'échelle globale, et dans la mesure du possible, c'est à cette échelle que l'on souhaiterait résoudre (2.1). Cependant, compte tenu de l'écart entre les deux échelles, on suppose qu'une telle résolution est techniquement impossible, du moins par une méthode numérique classique. On va donc présenter des méthodes numériques permettant d'obtenir une approximation de la solution fine, à partir de calcul à une échelle intermédiaire, dite *grossière*.

Comme l'illustre la Figure 2.1, les méthodes ici présentées s'appuient sur deux discrétisations des quantités du problème (2.1) : une *fine*, de taille caractéristique h, et une *grossière*, de taille caractéristique H, avec $h<H$. Dans tout ce qui suit, on indice donc d'un h les grandeurs numériques liées à l'échelle fine, microscopique.

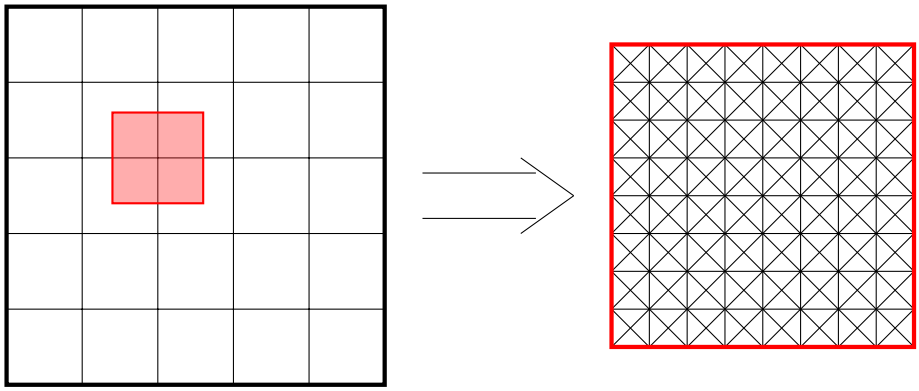


FIGURE 2.1 – Le domaine Ω est discrétisé grossièrement par le maillage $T_H(\Omega)$ (à gauche). On fait intervenir un maillage fin $T_h(\hat{K})$ (à droite) pour obtenir des informations locales sur une zone de travail précise $\hat{K}$ (en rouge).

et d'un H les grandeurs liées à l'échelle grossière, macroscopique. On note ainsi $T_H(\Omega)$ une discrétisation grossière du milieu Ω, respectivement $T_h(\Omega)$ une discrétisation fine. De même, un problème numérique résolu sur $T_H(\Omega)$, respectivement $T_h(\Omega)$, est dit *grossier*, respectivement *fin*.

On répartit les méthodes de simulations multi-échelles en trois grandes catégories les *méthodes de Galerkin directes*, qui comprennent les méthodes développées au cours de ces travaux de thèse, les *méthodes de Galerkin approchées* et les *méthodes de Volumes Finis*.

2.1 Présentation des méthodes multi-échelles.

On considère une discrétisation grossière $T_H(\Omega)$ du domaine de travail Ω. On appelle *méthode de simulation multi-échelle* toute méthode numérique respectant le schéma algorithmique décrit par l'Algorithme 2.1 :

Algorithme 2.1 Algorithme d'une méthode de simulation multi-échelle.

1: Choix d'un schéma grossier sur $T_H(\Omega)$.
2: Calcul, à l'échelle fine, des informations nécessaires au schéma grossier.
3: Résolution du problème grossier.
4: Reconstruction d'une solution $C_{H,h}$ à l'échelle fine.

La seconde étape de l'Algorithme 2.1 (Alg. 2.1 L.2) est la clé de toute méthode de simulation multi-échelle. Il s'agit en effet de calculer les éléments nécessaires à l'assemblage du problème grossier ; comme les matrices de rigidité dans le cas des méthodes Éléments Finis, ou des transmibilités dans le cas des méthodes Volumes Finis ; en y intégrant autant que possible les informations dont on dispose à

l'échelle fine. Pour ce faire, on résout généralement une série de *problèmes fins* sur des parties du domaine Ω. Ces problèmes doivent eux aussi être discrétisés et résolus, le choix de la méthode numérique étant généralement libre. Très souvent, ces problèmes sont indépendants les uns des autres et on peut les résoudre en parallèle.

La dernière étape (Alg. 2.1 L.4) de la méthode consiste à reconstruire une solution à l'échelle fine $C_{H,h}$ à partir de la solution grossière C_H. Cette étape n'est souvent appliquée qu'à une portion restreinte du domaine Ω, pour des raisons de taille mémoire, les méthodes multi-échelles ayant justement été développées pour résoudre des problèmes de très grandes tailles. Selon la méthode, elle peut être très simple, ou au contraire demander de résoudre un nouveau problème fin.

Cette définition regroupe un ensemble assez disparate de méthodes, que l'on peut répartir en trois grandes catégories : les méthodes de Galerkin directes, les méthodes de Galerkin approchées et les méthodes de Volumes Finis.

Les deux premières catégories de méthodes permettent de résoudre, à l'échelle grossière, une forme faible du problème (2.1). Il s'agit alors de résoudre un problème linéaire de Galerkin de la forme :

$$\begin{array}{l} \text{Trouver } C_H \in V_H \text{ tel que :} \\ \forall v_H \in V_H \quad B_H(C_H, v_H) \;=\; a_H(v_H) \end{array} \tag{2.2}$$

où V_H est un espace vectoriel «bien choisi», B_H une application bilinéaire symétrique de $V_H \times V_H$ dans $\mathbb{R}$, et a_H une application linéaire de V_H dans $\mathbb{R}$.

La manière de résoudre le problème (2.2) diffère d'une catégorie à l'autre. Dans le cas des *méthodes de Galerkin directes*, on cherche à déterminer V_H, B_H et a_H de manière à ce qu'ils tiennent compte des informations disponibles à l'échelle fine. Cela passe généralement par la détermination d'une famille de fonctions $(\Phi^I)_{I\in\mathscr{I}}$, solutions de *problèmes de cellules*, qui sert de base à l'espace V_H.

Au contraire, les *méthodes de Galerkin approchées* s'appuient sur un schéma grossier, et donc un triplet (V_H, B_H, a_H) *a priori* indépendant de l'échelle fine. On peut choisir, par exemple, une méthode Éléments Finis d'ordre 1 classique. L'utilisation de ce schéma grossier nécessite cependant le calcul et l'assemblage de termes matriciels qui, eux, sont liés aux données fines du problème (2.1), notamment la diffusivité D. Dans une méthode de Galerkin approchées, ces termes sont alors approximés à partir de *solutions locales* du problème (2.1).

Comme leur nom l'indique, les méthodes de *Volumes Finis multi-échelles* permettent de résoudre le problème (2.1) par une méthode de Volumes Finis. Dans ce cas, l'assemblage du problème numérique correspondant nécessite de calculer des *transmissibilités*, c'est-à-dire d'exprimer des flux à travers les arêtes du maillage grossier $T_H(\Omega)$. Ceux-ci seront exprimées à l'aide des solutions de *problèmes de cellules* définis sur le maillage dual $\tilde{T}_H(\Omega)$ du maillage $T_H(\Omega)$.

2.2 Méthodes de Galerkin directes.

2.2.1 Présentation de la méthode.

Cette catégorie de méthodes multi-échelles est la plus ancienne historiquement parlant, puisqu'elle comprend la première méthode multi-échelle, construite à partir d'une méthode d'Éléments Finis. Elle fut introduite par deux publications : tout d'abord *Hou et Wu* [120], qui présenta la méthode numérique ainsi que quelques exemples d'applications, puis *Hou et al* [121] qui en établit la convergence théorique.

Le principe de la méthode est relativement simple, puisqu'il s'agit de résoudre le problème grossier (2.1) par une méthode d'Éléments Finis d'ordre 1. Cependant, la base d'Éléments Finis $(\Phi^I)_{1\leq I\leq p}$ n'est pas composée de fonctions analytiques, mais à partir de solutions de *problèmes de cellules*.

Sur chaque macroélément $K \in T_H(\Omega)$, on résout donc :

$$-\nabla\cdot(D\nabla\Phi_K^I) \quad = \quad 0 \tag{2.3}$$

Le problème (2.3) nécessite des conditions aux limites sur ∂K pour être bien posé. On choisit d'imposer à Φ_K^I d'être linéaire sur ∂K, et de prendre pour valeur 1 en le noeud I, et 0 aux autres noeuds. Ces problèmes sont résolus numériquement sur des maillages fins $T_h(K)$. La fonction Φ^I est ensuite définie morceaux par morceaux à partir des fonctions Φ_K^I correspondantes. Dans le cas d'une discrétisation grossière en rectangle, on doit ainsi résoudre quatre problèmes de cellules par noeud du maillage $T_H(\Omega)$.

Une fois la famille $(\Phi^I)_{1\leq I\leq p}$ obtenue, on peut assembler la méthode d'Éléments Finis. Si on s'affranchit des conditions aux limites du problème général (2.1), on a :

$$V_H = Vect\{(\Phi^I)_{1\leq I\leq p}\} \tag{2.4}$$

$$B(u_H, v_H) = \int_\Omega D\nabla u_H\cdot\nabla v_H \tag{2.5}$$

$$a(v_h) = \int_\Omega f v_H \tag{2.6}$$

On résout donc numériquement le problème (2.2) afin d'obtenir une solution $\mathbf{C}_H$, un vecteur disposant d'une valeur par sommet I de $T_H(\Omega)$.

La construction à l'échelle fine de $C_{H,h}$ ne demande pas de résoudre un problème supplémentaire. En effet, pour obtenir $C_{H,h}$, il suffit de pondérer les fonctions $(\Phi^I)_{1\leq I\leq p}$ par les valeurs correspondantes de $\mathbf{C}_H$:

$$C_{H,h}(x) = \sum_{1\leq I\leq p} \mathbf{C}_H(I)\Phi^I(x) \tag{2.7}$$

Cette construction demande cependant une manipulation informatique des fonctions fines $\{\Phi_K^I\}$ pour réaliser les opérations de sommation et de multiplication. À cela s'ajoute un problème d'interpolation entre maillages si, pour un même macroélément K, les fonctions $\{\Phi_K^I\}$ ont été résolues sur des maillages fins $T_h(K)$ différents.

On trouvera à la Figure 2.2 (en haut) une illustration de l'ensemble des problèmes fins qu'il est nécessaire de résoudre pour utiliser cette méthode. En bleu autour du sommet I apparaissent les quatre problèmes de cellules. Pour obtenir la solution sur l'élément K (trait gras rouge), aucun problème supplémentaire n'est nécessaire.

2.2.2 Améliorations.

Depuis son introduction, la méthode d'Éléments Finis multi-échelle de *Hou et Wu* a fait l'objet de plusieurs améliorations. Sur le plan théorique notamment, elle a été intégrée à un cadre plus général de méthodes d'Éléments Finis multi-échelles d'ordre quelconque par les travaux de *Babuška et al.* [33, 32] et *Matache et al.* [140]. Deux sources d'erreurs ont de plus été identifiées et corrigées.

La première et plus importante, est un phénomène de *couche limite*. En effet, compte tenu des conditions aux limites imposées aux problèmes de cellules (2.3), on impose à la solution $C_H \in V_H$ d'être linéaire sur les frontières des macroéléments $K \in T_H(\Omega)$, ce que n'est pas C. Pour résoudre ce problème, une méthode de suréchantillonnage a été proposée dès l'article originel de *Hou et Wu*, même si son apport réel n'a pu être quantifié que quelques années plus tard [90].

Une autre méthode consiste à remplacer les conditions linéaires par des conditions *oscillantes* [28, 120]. Le principe est d'utiliser la solution d'un problème de cellule à la dimension $d-1$ comme conditions aux limites des problèmes de cellule en dimension d. Cette technique est très efficace, et on constate une disparition totale du phénomène de couche-limite. Elle est cependant difficile à mettre en œuvre, surtout dans un contexte $3D$, puisqu'elle nécessite alors de résoudre, non pas un problème par sommet, mais un problème par arête, par face et par sommets de K.

La seconde source d'erreur est un problème de *résonance de cellule*, qui apparaît dès que $K \in T_H(\Omega)$ n'est pas une cellule de périodicité du problème. Pour limiter cette erreur, *Hou et al.* [119] ont introduit une méthode multi-échelle de Petrov-Galerkin. Dans cette méthode, l'espace des solutions $Vect\{(\Phi^I)_{1\leq I\leq p}\}$ reste inchangé, mais les fonctions tests v_h sont choisis dans un espace d'Éléments Finis classique.

Dans [28], *Allaire et Brizzi* proposent un nouveau jeu de problèmes de cellules. Sur chaque macroélément $K \in T_H(\Omega)$, on ne résout en fait que d problèmes, où d

est la dimension du domaine. Les fonctions $(\Phi^I)_{1\leq I\leq p}$ sont alors définis par composition d'une fonction d'Éléments Finis classique par les solutions des problèmes de cellules. De cette manière, le nombre de problèmes de cellules est indépendant de l'ordre de la méthode d'Éléments Finis multi-échelle choisie, au contraire de la méthode introduite par *Hou et Wu*. Plus récemment, *Owhadi et Zhang* [152, 153] ont utilisé une approche similaire, en utilisant cette fois-ci des *problèmes correcteurs* au lieu de problèmes de cellules.

2.2.3 Extensions et applications.

Les méthodes d'Éléments Finis multi-échelles ont été étendues à d'autres problèmes linéaires. Par exemple, *Efendief et Wu* ont présentée une analyse détaillée de la méthode des Éléments Finis multi-échelles pour un nombre d'échelles quelconque [92]. On applique dans *Hou et al.* une méthode d'Éléments Finis multi-échelles à un problème de saut aux interfaces [119].

Les méthodes de simulations multi-échelles ont été appliquées à la formulation mixte du problème (2.1) par *Chen et Hou* [69]. Il s'agit de déterminr (q,C) solutions de :

$$\left\{\begin{array}{rcll} \nabla\cdot q & = & f & \text{dans } \Omega, \\ q & = & -D\nabla C & \text{dans } \Omega. \end{array}\right. \tag{2.8}$$

On construit alors une base d'Éléments Finis mixtes multi-échelles en s'appuyant sur les méthodes d'Éléments Finis mixtes classiques [55, 156]. On obtient alors une méthode de Galerkin directe très similaire à la méthode originale de *Hou et Wu*. Les différences ne tiennent, au final, qu'au choix des termes sources et des conditions aux limites des problèmes de cellule. Les Éléments Finis mixtes multi-échelles ont notamment été appliqués au transport incompressible d'un mélange eau-huile à travers un milieu poreux par *Aarnes* [1] et *Aarnes* et al. [4].

En 2008, *Aarnes et al.* [2] ont étendu cette formulation à un cadre plus général permettant notamment l'étude du transport diphasique en milieu poreux. Une analyse mathématique rigoureuse y est décrite, prouvant la convergence de la méthode. On peut également citer l'existence d'une méthode multi-échelle de Galerkin discontinue, qui combine les travaux de *Aarnes et al.* sur les Éléments Finis mixtes et ceux de *Hou et Wu* sur les Éléments Finis [3].

La méthode des Éléments Finis multi-échelles a été appliquée au cas général des équations elliptiques non linéaires par *Efendiev et al.* dans [91]. Le problème à résoudre est alors :

$$-\nabla\cdot(D(x,C,\nabla C))+a(x,C,\nabla C)=f \text{ dans } \Omega \tag{2.9}$$

La méthode semble bien s'adapter aux non-linéarités du problème, et on observe les mêmes forces et faiblesses que dans la cadre linéaire. En particulier, on

peut montrer que le phénomène de *couche limite* est la plus grande source d'erreur de la méthode. Comme dans le cadre linéaire, une méthode de sur-échantillonnage permet cependant de réduire efficacement ce phénomène.

Enfin, on peut citer les travaux de *Chen* [67] puis *Chen et al.* [68] qui, par le biais de réécritures précises du problème de Galerkin (2.2), ont grandement amélioré les estimations d'erreurs liant $C_{H,h}$ et C, que la méthode soit conforme ou non.

2.3 Méthodes de Galerkin approchées.

On présente dans cette section les méthodes multi-échelles *de Galerkin approchées*, qui ont été introduites par *E et Engquist* sous le nom de *Heterogeneous multiscale methods* [86]. Deux composants articulent la méthode : d'une part la sélection d'un solveur macroscopique, correspondant à l'étape (Alg. 2.1 L.1), et d'autre part l'estimation des données nécessaires à ce schéma par la résolution de problèmes locaux, correspondant à l'étape (Alg. 2.1 L.2). Contrairement aux méthodes de Galerkin directes, le choix de la méthode macroscopique ne dépend pas des informations dont on dispose à l'échelle fine. C'est la mise en œuvre de cette méthode, c'est-à dire le calcul de termes matriciels ou de flux à l'étape (Alg. 2.1 L.2), qui fait intervenir l'échelle fine du problème.

La méthodologie multi-échelle hétérogène présentée par *E et Engquist* est en fait une méthodologie générale permettant de construire des algorithmes multi-échelles économes. Les échelles du problème étant séparées, on peut notamment travailler sur des modèles différents d'une échelle à l'autre.

2.3.1 Présentation de la méthode.

Dans le cadre du problème de Galerkin (2.2), la méthode proposée par *E et al.* dans [87] s'appuie, à l'échelle grossière, sur une méthode d'Éléments Finis d'ordre 1. On note $(\Phi^I)_{1\leq I\leq p}$ la base d'Éléments Finis correspondante.

Si on s'affranchit des conditions aux limites, le problème (2.2) est défini par :

$$V_H = Vect\{(\Phi^I)_{1\leq I\leq p}\} \tag{2.10}$$

$$B(u_H, v_H) = \int_\Omega D_H \nabla u_H \cdot \nabla v_H \tag{2.11}$$

$$a(v_h) = \int_\Omega f v_H \tag{2.12}$$

où D_H représente la valeur effective de D à l'échelle grossière. En effet, on ne peut utiliser ici la valeur fine D_h, puisque le schéma grossier doit être indépendant de l'échelle fine.

Afin d'assembler le schéma grossier, il est nécessaire de calculer les différents termes $(B(\Phi^I, \Phi^J))_{1\leq I,J\leq p}$ de la matrice de rigidité. Ces termes peuvent être évalués

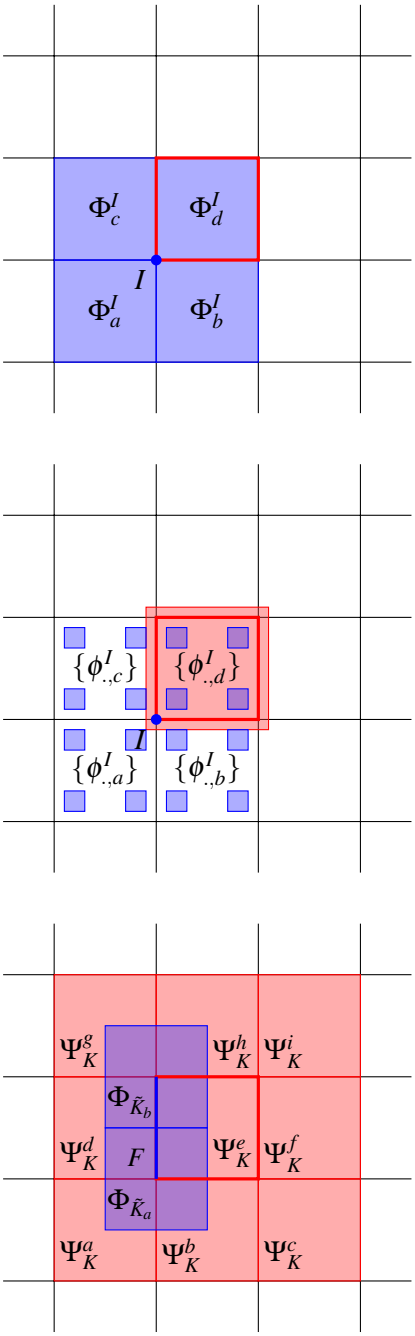


FIGURE 2.2 – Illustration des méthodes de Galerkin directes (en haut), de Galerkin approchées (au milieu) et de Volumes Finis multi-échelles (en bas) sur un maillage grossier. En bleu, les problèmes fins relatifs au noeud I ou à la face F, selon la méthode. En rouge, les problèmes à résoudre *a posteriori* pour obtenir une solution fine sur l'élément grossier K (en rouge, trait épais).

par une formule de quadrature. Soit $K \in T_H(\Omega)$ un élément du maillage grossier $T_H(\Omega)$, de volume $|K|$. On note x_m les points de quadratures appartenant à K et p_m les poids associés. On a alors :

$$B(\Phi^I, \Phi^J) = \int_\Omega D\nabla\Phi^I \cdot \nabla\Phi^J$$
$$B(\Phi^I, \Phi^J) \approx \sum_{K \in T_H(\Omega)} |K| \sum_{x_m \in K} p_m (D_H \nabla\Phi^I \cdot \nabla\Phi^J)(x_m)$$

D_H étant une quantité grossière, on ne dispose pas de sa valeur en x_m. On va donc déterminer une valeur de $(D_H\nabla\Phi^I \cdot \nabla\Phi^J)$ en x_m par le biais de problèmes locaux.

Soit $\omega(x_m) \subset K$ un domaine cubique centré en x_m, on résout pour chaque Φ^I le problème suivant :

$$\left\{ \begin{array}{rcll} -\nabla(D_h \nabla\phi_m^I) & = & 0 & \text{sur } \omega(x_m), \\ \phi_m^I & = & \Phi^I & \text{sur } \partial\omega(x_m). \end{array} \right. \tag{2.13}$$

Le terme $(D_H\nabla\Phi^I \cdot \nabla\Phi^J)$ est alors calculé par moyenne de la quantité fine correspondante :

$$(D_H \nabla\Phi^I \cdot \nabla\Phi^J)(x_m) \approx \frac{1}{|\omega(x_m)|} \int_{\omega(x_m)} D_h \nabla\phi_m^I \cdot \nabla\phi_m^J$$

Une fois les termes $(B(\Phi^I, \Phi^J))_{1 \leq I,J \leq p}$ calculés, on peut assembler et résoudre le problème matricielle grossier, afin d'obtenir la solution grossière C_H.

Soit $\Omega_h \subset \Omega_\mu$ deux sous-domaines de Ω, vérifiant $dist(\partial\Omega_h, \partial\Omega_\mu) > 0$. Si l'on souhaite obtenir une valeur fine $C_{H,h}$ sur le domaine Ω_h, la méthode de *E et al.* propose de résoudre :

$$\left\{ \begin{array}{rcll} -\nabla(D_h \nabla C_{H,h}) & = & f & \text{sur } \Omega_\mu, \\ C_{H,h} & = & C_H & \text{sur } \partial\Omega_\mu. \end{array} \right. \tag{2.14}$$

On restreint ensuite la solution $C_{H,h}$ au seul domaine Ω_h. Le problème (2.14) n'est pas résolu directement sur Ω_h afin de s'affranchir d'un phénomène de *couche limite*.

On illustre à la Figure 2.2 (au milieu) l'ensemble des problèmes fins qu'il est nécessaire de résoudre pour utiliser cette méthode. En bleu autour du sommet I apparaissent les 4×4 problèmes fins utilisés pour calculer les $\{\phi_m^I\}$, chacun sur un domaine $\omega(x_m)$ distinct. Pour obtenir la solution sur l'élément K (trait gras rouge), il est nécessaire de résoudre le problème (2.14) sur un domaine légèrement plus grand (en rouge).

2.3.2 Extensions et applications.

De nombreuses analyses théoriques ont été réalisées sur la méthode proposée par *E et al.* [86, 87].

Outre les éléments présentés dans l'article originel, on peut citer par exemple les travaux de *Abdulle* [15]. Celui-ci analyse l'influence des problèmes locaux (2.13) sur la méthode. Il présente notamment des estimations d'erreurs *a priori* liant la taille des domaines $\omega(x_m)$ et les erreurs de résolution de (2.13) à la convergence de la méthode.

Ohlberger [151] propose une interprétation du problème de Galerkin approchée (2.10-2.12) comme la formulation variationnelle d'un problème homogénéisé limite, au sens de la convergence double-échelle [26]. Cette nouvelle formulation permet de retrouver les estimations d'erreurs *a priori* de *E et al.*et de *Abdulle*, mais aussi de construire des estimations d'erreurs *a posteriori*. Les travaux d'*Ohlberger* s'appuie cependant sur des hypothèses fortes de régularité, et des recherches pour déterminer des estimations d'erreurs *a posteriori* dans le cas général sont en cours [17].

L'influence des conditions limites imposées aux problèmes locaux (2.13) a fait l'objet de travaux par *Yu et E* [175]. Trois types de conditions limites sont étudiées : les conditions de Dirichlet, les conditions de Neumann et les conditions périodiques. À partir de plusieurs simulations numériques, les auteurs concluent que les conditions périodiques sont plus efficaces. Les conditions de Dirichlet, respectivement de Neumann, semblent surestimer, respectivement sous-estimer, la diffusivité effective.

La méthode de *E et al.* a également fait l'objet d'adaptation afin de pouvoir traiter une plus large gamme de problèmes. Selon l'article originel de *E et al.*, l'étude des problèmes elliptiques non linéaires ne pose pas de difficultés, :

$$-\nabla \cdot (D(x,C)\nabla C) = f \text{ dans } \Omega$$

La justification théorique et les estimations d'erreurs *a priori* de la méthode demandent cependant un travail particulier [87].

Dans [18], *Abdulle et Schwab* cherchent à résoudre le problème (2.1) sur un domaine Ω dont la surface comprend de fortes oscillations à l'échelle fine. Par le biais de transformations géométriques, les auteurs se ramènent à un problème sur un domaine de référence, en transférant les perturbations fines sur le terme source f. En plus de B_H, il s'agit donc de construire une approximation de a_H et la méthode de Galerkin approchée est adaptée en conséquences.

On peut également citer une adaptation de la méthode au cas des problèmes d'homogénéisation paraboliques [144], et une méthode multi-échelle s'appuyant sur une méthode de Galerkin discontinue [16].

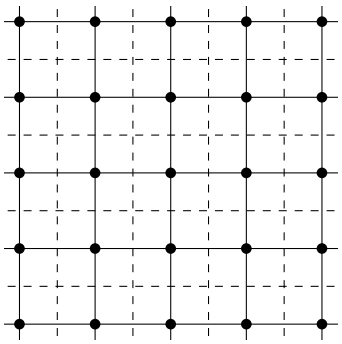


FIGURE 2.3 – Le maillage dual $\tilde{T}_H(\Omega)$ (tiret) est construit à partir des barycentres des éléments du maillage primal $T_H(\Omega)$ (trait continu). Chaque élément du maillage dual contient un unique noeud de $T_H(\Omega)$ (points noir).

2.4 Méthodes des Volumes Finis multi-échelles.

2.4.1 Présentation de la méthode.

La méthode de Volumes Finis multi-échelles fut introduite par *Jen et al.* [125], dans le cadre d'un travail sur le transport en milieu poreux ; plus précisément celui d'eau et d'huile au sein d'un réservoir pétrolifère. Il s'agissait de résoudre numériquement le problème (2.1) tout en s'assurant de calculer des flux conservatifs, d'où l'idée de développer une méthode multi-échelle s'appuyant sur les Volumes Finis. Outre la solution C, il était également important de pouvoir disposer d'une valeur précise du champs $-D\cdot\nabla C$. Ce sont en effet deux grandeurs essentielles, sous les noms de *pression* et *vitesse*, aux simulations de l'industrie pétrolière. Les aspects théoriques ont été étudiés par *Ginting* [105].

Afin de présenter la méthode des Volumes Finis multi-échelles, il est nécessaire d'introduire la formule d'équilibre des flux. Pour tout élément $K \in T_H(\Omega)$, on a :

$$\sum_{F\in\partial K}\int_F -D\nabla C\cdot\mathbf{n}_K = \int_F f \tag{2.15}$$

où $\mathbf{n}_K$ est le vecteur normal à la face F, sortant de K.

On suppose C_H constant par morceaux sur chaque élément $K \in T_H(\Omega)$. Le principe des méthodes de Volumes Finis est de construire un schéma numérique en partant des propriétés du flux $L_{K,F}$ de C à travers une face F du maillage $T_H(\Omega)$.

$$L_{K,F} = \int_F (D\nabla C\cdot\mathbf{n}_K) \tag{2.16}$$

Afin d'approximer $L_{K,F}$ en fonction des $\mathbf{C}_H(K)$, on va faire intervenir $\tilde{T}_H(\Omega)$ le maillage dual de $T_H(\Omega)$. Comme le montre la Figure 2.3, les noeuds de ce maillage sont les barycentres des éléments du maillage $T_H(\Omega)$. Deux noeuds de $\tilde{T}_H(\Omega)$ sont

reliés si les éléments correspondant de $T_H(\Omega)$ sont voisins. Construit de cette manière, chaque élément $\tilde{K}$ du maillage dual $\tilde{T}_H(\Omega)$ contient un noeud de $T_H(\Omega)$. On suppose alors que, sur chaque élément $\tilde{K} \in \tilde{T}_H(\Omega)$, on a :

$$C = \sum_{K, K\cap\tilde{K}\neq\emptyset} \mathbf{C}_H(K)\Phi^K_{\tilde{K}} \tag{2.17}$$

Pour une méthode Volumes Finis classique, les $(\Phi^K_{\tilde{K}})$ sont des fonctions analytiques. Dans le cas d'une méthode multi-échelle, on les choisit solutions des *problèmes de cellules duaux* :

$$-\nabla\cdot(D\nabla\Phi^K_{\tilde{K}}) \quad = 0 \quad \text{dans } \tilde{K}. \tag{2.18}$$

Les conditions aux limites du problème (2.18) sont celles employées par la méthode Éléments Finis multi-échelles de *Hou et Wu* : *linéaires* ou *oscillantes*. Dans le cas des conditions limites *linéaires*, cela signifie qu'on impose à $\Phi^K_{\tilde{K}}$ d'être linéaire sur $\partial\tilde{K}$ et d'avoir pour valeurs 1 en le sommet correspondant au barycentre de l'élément K, et 0 en les autres sommets. Pour plus de détails à ce sujet, on peut se reporter à [125] .

Ces problèmes sont résolus numériquement à l'échelle fine, sur des maillages fins $T_h(\hat{K})$. Les équations (2.15), (2.16) et (2.17) permettent ensuite d'assembler un système linéaire liant les valeurs de $\mathbf{C}_H$.

Une fois obtenue la solution grossière C_H, il est possible d'utiliser la base duale $(\Phi^K)_{K\in T_H(\Omega)}$ pour reconstruire une solution fine $C_{H,h}$.

$$C_{H,h}(x) = \sum_{K\in T_H(\Omega)} C_H(K)\Phi^K(x) \tag{2.19}$$

Cepenant, si l'on procède ainsi, la vitesse $D\nabla C_{H,h}$ ainsi obtenue est discontinue le long des frontières ∂K, et le champs n'est plus à divergence nulle.

Pour remédier à cela, on introduit un second jeu de problèmes locaux. Pour chaque $K \in T_H(\Omega)$, on résout sur chaque élément K_1 partageant un sommet avec K, le problème suivant :

$$-\nabla\cdot(D\nabla\Psi^{K_1}_K) = f_K \qquad \text{dans } K_1, \tag{2.20}$$

$$\Psi^{K_1}_K = \sum_{\tilde{K}, K_1\cap\tilde{K}\neq\emptyset} \Phi^{K_1}_{\tilde{K}} \qquad \text{sur } \partial K_1 \tag{2.21}$$

On exprime alors $C_{H,h}$ sur chaque macroélement $K \in T_H(\Omega)$ par :

$$C_{H,h|K} = \sum_{K_1} \mathbf{C}_H(K_1)\Psi^{K_1}_K \tag{2.22}$$

On trouve à la Figure 2.2 (en bas) une illustration de l'ensemble des problèmes fins qu'il est nécessaire de résoudre pour utiliser cette méthode. En bleu autour de la face F apparaissent les deux problèmes de cellules duaux (2.18), chaque problème portant sur une maille du maillage dual $\tilde{T}_H(\Omega)$. Pour obtenir la solution sur l'élément K (trait gras rouge), il est nécessaire.de résoudre neuf problèmes (2.21), un par élément jouxtant K (en rouge).

2.4.2 Extensions et applications.

La problématique du transport diphasique eau-huile fut la première extension de la méthode des Volumes Finis multi-échelles [126, 127]. La méthode n'intervient en fait que sur une partie du problème, afin de résoudre l'équation de *pression*. La seconde moitié du problème, l'équation de *transport*, est résolue à l'échelle fine explicitement [126] ou implicitement [127].

L'équation de pression est similaire à l'équation (2.1), mais la *mobilité D* est dans ce cadre une fonction de la saturation. Elle évolue donc dans le temps avec la répartition des différentes phases et il devient nécessaire de recalculer les problèmes locaux correspondants, duaux et fins, à chaque pas de temps. En pratique, ce recalcul est restreint aux zones où la mobilité a été modifiée significativement. Les seuils sont définis de manière arbitraire, en accord avec la physique du problème à résoudre. Dans le cas d'un transport diphasique, modélisant le déplacement de pétrole suite à l'injection d'eau, il suffit de concentrer le recalcul sur les fronts de saturation, et de laisser le reste du domaine en l'état.

Une version itérative de la méthode des Volumes Finis multi-échelles a été récemment développée, afin d'améliorer ses résultats sur les milieux fortement hétérogènes et anisotropes [111]. En lieu et place de (2.17), on calcule itérativement :

$$C^t_{|\tilde{K}} = \sum_{K, K\cap\tilde{K}\neq\emptyset} \mathbf{C}_H(K)\Phi^K_{\tilde{K}} + \Theta^t_{\tilde{K}} \tag{2.23}$$

où $\Theta^t_{\tilde{K}}$ est un terme de correction calculée sur chaque volume $\tilde{K_j}$ du maillage dual $\tilde{T_H(\Omega)}$.

$$-\nabla\cdot(D\nabla\Theta^{t+1}_{\tilde{K}}) = \mathscr{R}(\mathscr{S}^{n_t}(C^t)) \tag{2.24}$$

$\mathscr{R}$ est un opérateur différentiel qui découle naturellement de l'injection de (2.23) dans (2.1). $\mathscr{S}$ est une fonction de lissage, que l'on applique un nombre n_t de fois à la fonction C^t. Son rôle est fondamental dans la convergence de la méthode. Utilisée correctement, elle permet en effet de distribuer dans $\tilde{K_j}$ l'erreur de la solution habituellement concentrée le long des frontières $\partial\tilde{K}$.

Cette méthode permet de s'affranchir du calcul des problèmes locaux (2.21), le terme de correction $\Theta^t_{\tilde{K}}$ permettant d'assurer la continuité de C et de $D\nabla C$. On se reportera à [111] pour plus de détails sur ce point.

D'autres améliorations ont depuis été introduites, afin de prendre en compte les différents aspects du transport d'un ou plusieurs fluides au sein d'un réservoir pétrolifère. Sur le plan phénoménologique, on peut par exemple citer la prise en compte de la compressibilité des fluides [134, 137] et de la *capillarité* [138]. D'un point de vue plus pratique, le cas de répartitions complexes des termes sources ; modélisant puits, injecteurs, etc ; a fait l'objet d'études récentes [128, 174].

Chapitre 3

Méthodes multi-échelles pour les matériaux cimentaires.

Introduction.

On présente dans ce chapitre les méthodes multi-échelles qui ont été développées durant ce travail de thèse. Elles sont au nombre de trois et appartiennent à la catégorie des *méthodes multi-échelles de Galerkin directes*, telle que caractérisée à la section §2.2.

Dans tout ce qui suit, Ω est soit un domaine rectangulaire (dimension $d = 2$) ou parallélépipédique (dimension $d = 3$). On cherche à résoudre un problème de diffusion sur Ω.

$$\left\{\begin{array}{rcll} -\nabla\cdot(D\nabla C) & = & f & \text{dans } \Omega, \\ C & = & g_D & \text{sur } \Gamma_D, \\ D\nabla C\cdot\mathbf{n} & = & g_N & \text{sur } \Gamma_N, \end{array}\right. \tag{3.1}$$

où (Γ_D, Γ_N) est une partition de $\Gamma = \partial\Omega$ la frontière du domaine Ω. f, g_D et g_N sont respectivement le terme source, la condition aux limites de Dirichlet et la condition aux limites de Neumann.

Ce problème est approché par un problème linéaire de Galerkin, que l'on appelle *problème grossier* et qui prend la forme suivante :

$$\begin{array}{l} \text{Trouver } C_H \in V_H \text{ tel que :} \\ \forall v_H \in V_H \; B_H(C_H, v_H) = a_H(v_H) \end{array} \tag{3.2}$$

où V_H est un espace vectoriel «bien choisi», B_H une application bilinéaire symétrique de $V_H \times V_H$ dans $\mathbb{R}$, et a_H une application linéaire de V_H dans $\mathbb{R}$.

Chacune des méthodes présentées suit le processus multi-échelle décrit à l'Algorithme 3.1, qui reprend les étapes de l'Algorithme 2.1. Le domaine de travail est tout d'abord discrétisé par le biais d'un maillage grossier $T_H(\Omega)$ (Alg. 3.1 L.1).

Ensuite, afin de résoudre le problème (3.2), on détermine l'espace vectoriel V_H qui est construit à partir des solutions des problèmes de *cellules*. Il s'agit de problèmes particuliers de diffusion définis sur chaque élément $K \in T_H(\Omega)$, que l'on résout sur un maillage fin (Alg. 3.1 L.2). Le problème grossier est ensuite assemblé, une opération qui demande de calculer plusieurs termes à l'échelle fine, et résolu (Alg. 3.1 L.3). Enfin, on calcule une solution fine $C_{H,h}$ à partir de la solution grossière C_H de (3.2) (Alg. 3.1 L.4).

Algorithme 3.1 Étapes des méthodes multi-échelles.

1: Découpage du domaine Ω.
2: Résolution des problèmes de cellules. ▷ Alg. 2.1 L.2
3: Construction et résolution du problème grossier sur V_H. ▷ Alg. 2.1 L.3
4: Reconstruction d'une solution $C_{H,h}$ à l'échelle fine. ▷ Alg. 2.1 L.4

La spécificité des méthodes ici présentées tient en partie dans la résolution des problèmes de cellules. En effet, on utilise ici des méthodes Volumes Finis pour résoudre tous les problèmes fins, alors que ceux-ci sont généralement résolus, dans ce contexte multi-échelle, par des méthodes de Galerkin. En effet, coupler deux types différents de méthodes aux échelles fine et grossière introduit des difficultés supplémentaires, ne serait-ce que pour construire le schéma grossier. Ce choix a cependant été motivé par plusieurs études sur les matériaux cimentaires [5, 47, 96] qui ont montré la supériorité, notamment en termes de stabilité, des méthodes Volumes Finis sur les problèmes à forte anisotropie et forts sauts de diffusivité.

De plus, des méthodes d'Éléments Finis *conformes*, d'Éléments Finis *non-conformes* et de Galerkin discontinues sont utilisées à l'échelle grossière. Si les deux premières ont déjà fait l'objet de nombreux travaux [28, 120, 121], l'utilisation dans ce contexte d'une méthode de Galerkin discontinue est, à notre connaissance, inédite.

On décrit, dans la suite du chapitre, chaque étape de l'Algorithme 3.1. Les méthodes numériques utilisées sont donc présentées au moment où elles interviennent dans le processus multi-échelle, c'est-à-dire lors de la résolution des problèmes locaux pour les méthodes Volumes Finis, et lors de la construction du schéma grossier pour les méthodes de Galerkin.

La première méthode développée utilise un schéma de Volumes Finis à neuf points au niveau fin, et un schéma Éléments Finis Q_1 à l'échelle grossière. Cette méthode a fait l'objet d'une implémentation sur une maquette indépendante, dont on présentera l'architecture et les résultats numériques au chapitre §4.

Les seconde et troisième méthodes utilisent un schéma de Volumes Finis dit *VFDiam* au niveau fin. Au niveau grossier, il s'agit respectivement d'un schéma d'Éléments Finis Q_1 et d'une méthode de Galerkin Discontinue à pénalité intérieure

[82]. Elles ont été implémentées au sein du code de calcul *MPCube* [61]. On en présente les résultats au chapitre §5.

La dernière partie §3.6 de ce chapitre est consacrée aux matériaux cimentaires. On y présente le modèle, dit *par inclusions*, choisi pour décrire les matériaux cimentaires au cours de ces travaux de thèse. La modélisation des phénomènes de dégradation, de vieillissement, est également présentée.

3.1 Découpage du domaine Ω.

Définition 3.1 *On appelle* division du milieu Ω *; et on note* $\mathscr{D}$ *; un* $d-$*uplet d'entiers strictement positifs. Le domaine* Ω *est alors uniformément divisé en* $\mathscr{D}(i)$ *éléments parallélépipédiques selon chaque direction i. Cette division construit donc une discrétisation grossière* $T_H(\Omega)$ *de* Ω*.*

Définition 3.2 *On appelle* macroélément *; et on note K ; un élément de la discrétisation grossière* $T_H(\Omega)$*. En appliquant l'ordre lexicographique aux barycentres des éléments de* $T_H(\Omega)$*, chaque macroélément K se voit attribuer un* $d-$*uplet d'entiers positifs. Ce* $d-$*uplet représente les coordonnées de K au sein de* $T_H(\Omega)$*. Généralement, on assimile le macroélément à ses coordonnées, que l'on note tous deux K.*

Dans le cas $3D$, chaque macroélément K est un parallélépipède rectangle dont les arêtes suivent les axes des coordonnées. En posant $m_K = (x_m, y_m, z_m)$ et $M_K = (x_M, y_M, z_M)$, on a l'égalité suivante :

$$K =]x_m, x_M[\times]y_m, y_M[\times]z_m, z_M[. \tag{3.3}$$

Définition 3.3 *On définit un ordre partiel* $\leq$ *sur les macroélements* K_a *et* K_b *par la relation suivante :*

$$K_a \leq K_b \qquad \Leftrightarrow \qquad \forall 1 \leq i \leq d \quad K_a(i) \leq K_b(i). \tag{3.4}$$

La Figure 3.1 (partie gauche) présente un exemple de division du domaine Ω, pour lequel on a choisi $\mathscr{D} = (2,2)$.

Définition 3.4 *On appelle* taux de sur-échantillonnage *; et on note* ρ *; un réel positif. On appelle* découpage *le couple* $(\mathscr{D}, \rho)$*. On appelle* cellule *; et on note* $\hat{K}$ *; l'union du macroélément K et d'une fraction* ρ *de ses voisins.*

Par définition, la cellule $\hat{K}$ empiète sur les macroéléments connexes de K, comme l'illustre la Figure 3.1 (droite). Dans le cadre $3D$, on a l'égalité suivante :

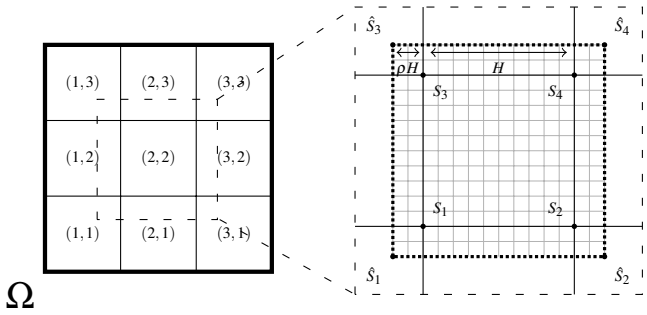


FIGURE 3.1 – Le domaine Ω (en gras) est divisé en macroélements rectangulaires K (ligne solide). La cellule $\hat{K}$ (en pointillé), construite à partir de K et d'une fraction ρ de ses voisins, est maillée finement (traits fins gris).

$$\begin{aligned}\hat{K} =]x_m - \rho(x_M - x_m), x_M + \rho(x_M - x_m)[\times \\ ]y_m - \rho(y_M - y_m), y_M + \rho(y_M - y_m)[\times \\ ]z_m - \rho(z_M - z_m), z_M + \rho(z_M - z_m)[. \quad (3.5)\end{aligned}$$

La méthode multi-échelle n'utilise pas nécessairement le sur-échantillonnage : on peut choisir $\rho = 0$ et retrouver alors $\hat{K} = K$.

Remarque 3.1 *Il est important de noter que, en pratique, pour des raisons de gestion de la mémoire, le milieu n'est jamais créé entièrement puis découpé. En effet, même si l'on dispose bien d'une description globale du milieu (diffusivités, positions et formes des inclusions, etc.), la construction du milieu (création des géométries, maillage) se fait directement cellule par cellule. Il est cependant parfaitement possible de fusionner tout ou partie des macroéléments pour obtenir une vision d'ensemble du milieu. Les détails pratiques de la construction des cellules (champs de diffusivités, génération de maillages, etc.) sont décrites à la section §4.3, pour la première méthode, et au chapitre §6, pour les deuxième et troisième méthodes.*

3.2 Résolution des problèmes de cellules.

3.2.1 Problèmes de cellules et Volumes Finis.

3.2.1.1 Problèmes de cellules.

Soit $n = 4$ (dimension 2) ou $n = 8$ (dimension 3), on appelle $(S_i)_{1\leq i\leq n}$ les *sommets* du macroélément K et $(\hat{S}_i)_{1\leq i\leq n}$ les sommets de la cellule $\hat{K}$ correspondante. Ces notations sont illustrées à la Figure 3.1 (droite).

Sur chaque cellule $\hat{K}$, pour chaque $1 \leq i \leq n$ on doit résoudre un *problème de cellule* :

$$\left\{ \begin{array}{rcll} -\nabla \cdot (D\nabla \Psi^i_{\hat{K}}) & = & 0 & \text{dans } \hat{K}, \\ \Psi^i_{\hat{K}} & = & \beta^i & \text{sur } \partial\hat{K}. \end{array} \right. \tag{3.6}$$

β^i est une fonction linéaire continue sur chaque face de $\partial\hat{K}$. En notant δ le symbole de Kronecker, on la définit univoquement par les conditions suivantes :

$$\forall 1 \leq j \leq n \qquad\qquad \beta^i(\hat{S}_j) = \delta_{i,j}. \tag{3.7}$$

Remarque 3.2 *Les conditions aux limites utilisées ici sont des conditions aux limites linéaires mais d'autres possibilités existent. En particulier, on présente dans [28, 120] l'impact positif de conditions* oscillantes *sur la méthode. Le principe est d'utiliser la solution d'un problème de diffusion à la dimension $d-1$ comme conditions aux limites des problèmes de cellule en dimension d. Ces conditions sont cependant difficiles à mettre en œuvre, surtout dans un contexte 3D, puisqu'elles nécessitent alors de résoudre, non pas un problème par sommet, mais un problème par arête, par face et par sommets de $\hat{K}$. Même si l'intérêt des conditions oscillantes est certains ; voir à ce sujet la section §3.3.1.1 ; on s'en tient donc, au cours de ces travaux, aux conditions linéaires.*

3.2.1.2 Principe des méthodes de Volumes Finis.

Pour résoudre les problèmes de cellule, deux méthodes numériques appartenant à la famille des Volumes Finis ont été utilisées : les méthodes *VF9* et *VFDiam*. On présente ici le principe général des méthodes de Volumes Finis, tandis que les spécificités des méthodes *VF9* et *VFDiam* sont l'objet des sections suivantes.

On quitte, pour le reste de la section, le cas particulier des problèmes de cellules, pour travailler sur un problème elliptique général. Soit Δ un domaine de $\mathbb{R}^d$ et soit à résoudre le problème elliptique suivant :

$$\left\{ \begin{array}{rcll} -\nabla \cdot (A\nabla u) & = & 0 & \text{dans } \Delta, \\ u & = & u_D & \text{sur } \partial\Delta. \end{array} \right. \tag{3.8}$$

Soit $T_h(\Delta)$ une discrétisation du domaine Δ. Pour tout élément $k \in T_h(\Delta)$, en utilisant la formule de Green, l'équation (3.8) peut s'écrire sous la forme intégrale suivante :

$$\sum_{f \in \partial k} \int_f -A\nabla u \cdot \mathbf{n}_f = 0 \tag{3.9}$$

où $\mathbf{n}_f$ est le vecteur normal à la face f, sortant de k.

Le principe des méthodes de Volumes Finis est de construire un schéma numérique en partant des propriétés du flux $L_{k,f}$ de u à travers une face f du maillage.

$$L_{k,f} = \int_f (A\nabla u \cdot \mathbf{n}_f) \tag{3.10}$$

Pour cela, on suppose généralement que u_h, la solution approchée de u, est constante sur chaque élément k de $T_h(\Delta)$. On réalise ensuite une approximation $\tilde{L}_{k,f}$ de $L_{k,f}$ à partir de points caractéristiques du milieu, comme les barycentres de k et f. Certaines des inconnues introduites par l'approximation sont supprimées en utilisant la continuité du flux à travers la face f :

$$\forall f \in \partial k_1 \cap \partial k_2 \qquad L_{k_1,f} + L_{k_2,f} = 0 \tag{3.11}$$

Pour chaque méthode des Volumes Finis, il faut donc présenter l'approximation $\tilde{L}_f$ utilisée. Il est également nécessaire, pour la suite du processus multi-échelle, de déterminer une valeur $u_h(f)$ pour chaque face f de $T_h(\Delta)$. Cette quantité n'étant pas naturellement calculée par les méthodes de Volumes Finis, on l'interpole à partir des valeurs connues de u_h.

Plus de détails sur les méthodes de Volumes Finis, ainsi qu'un grand nombre d'exemples, sont disponibles dans [95].

3.2.2 La méthode de Volumes Finis à neuf points *VF9*.

La première méthode que l'on présentera est la méthode de Volume Finis dite *à neuf points* ou *VF9* introduite par *Faille* [97]. Cette méthode s'applique dans un cadre bidimensionnel uniquement, et permet de résoudre des problèmes sur un maillage quadrangulaire irrégulier où A est un tenseur discontinu.

Comme on le voit dans la suite, l'approximation de L_f fait intervenir quatre éléments du maillage. Utilisée tour à tour sur les quatres cotés d'un élément du maillage, on lie ainsi les valeurs de neuf éléments, d'où le nom de la méthode. Si l'on suppose A diagonal, et que le maillage est rectangulaire «parallèle aux axes», cette méthode est équivalente à la méthode classique des *Volumes Finis à cinq points* [95].

3.2.2.1 Approximation du flux à travers une face.

On considère deux éléments (k^-, k^+) adjacents du maillage fin $T_h(\Delta)$, et on appelle f leur arête commune, comme l'illustre la Figure 3.2. On note G_f, h_f et $\mathbf{n}_f^+$ respectivement le barycentre, la longueur et la normale extérieure à E^+ de l'arête f.

On suppose la matrice A constante sur chaque élément, de valeurs respectives A^- et A^+. En notant ${}^tA$ la transposée de A, on a :

$$\begin{aligned} L_{k^+,f} &= \int_f (A^+ \nabla u \cdot \mathbf{n}_f^+) \\ &= \int_f (\nabla u \cdot {}^tA^+ \mathbf{n}_f^+) \end{aligned}$$

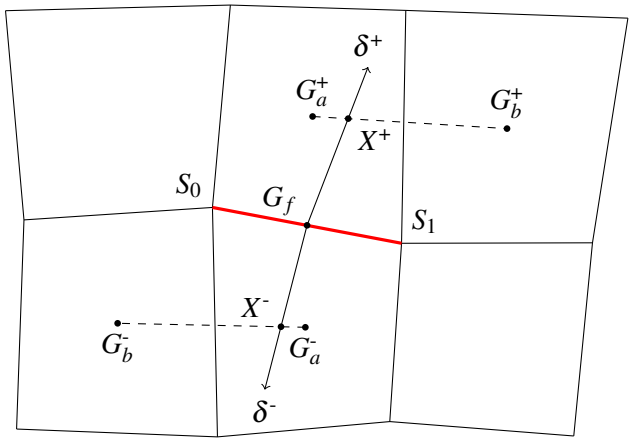


FIGURE 3.2 – Principaux points et droites introduits par le schéma *VF9* afin d'approximer le flux à travers la face $f = S_0S_1$ (en rouge).

.

Soit δ^+ la demi-droite d'origine G_f et de vecteur directeur $-{}^tA^+\mathbf{n}_f^+$. Parmi les éléments voisins de f, il existe alors deux éléments adjacents k_a^+ et k_b^+, de barycentres G_a^+ et G_b^+, tel que δ^+ coupe le segment $\left[G_a^+G_b^+\right]$. Nécessairement, k^+ est l'un de ces deux éléments.

On note X^+ le point d'intersection de δ^+ et de $\left[G_a^+G_b^+\right]$. On introduit u_{X^+} et u_{G_f} les valeurs de u en X^+ et G. On approxime alors le flux à travers f par :

$$\tilde{L}_{k^+,f} = -h_f a^+ \frac{u_{X^+} - u_{G_f}}{d_f^+} \tag{3.12}$$

où on a posé $a^+ = \|{}^tA^+\mathbf{n}_f^+\|_2$ la norme du vecteur ${}^tA^+\mathbf{n}_f^+$ et d_f^+ la distance entre les points G_f et X^+.On peut écrire une formulation similaire pour $L_{k^-,f}$. En utilisant (3.11), il vient alors :

$$u_{G_f} = \frac{d_f^- a^+ u_{X^+} + d_f^+ a^- u_{X^-}}{d_f^- a^+ + d_f^+ a^-} \tag{3.13}$$

On peut donc s'affranchir de u_{G_f} pour approximer $L_{k^+,f}$. Il ne reste plus qu'à exprimer u_{X^+} et u_{X^-} en fonction de valeurs aux barycentres des éléments de $T_h(\Delta)$. X^+ étant sur le segment $\left[G_a^+G_b^+\right]$, on note d_a^+ la distance entre G_a^+ et X^+, et on définit alors u_{X^+} comme une interpolation linéaire de u_a^+ et u_b^+ :

$$u_{X^+} = \frac{1}{d_a^+ + d_b^+}\left(d_a^+ u_a^+ + d_b^+ u_b^+\right) \tag{3.14}$$

En utilisant (3.13) et (3.14), on obtient finalement l'approximation :

$$\tilde{L}_{k^+,f} = -h_f \frac{a^+ a^-}{d_f^- a^+ + d_f^+ a^-} \left[\frac{1}{d_a^+ + d_b^+} (d_a^+ u_a^+ + d_b^+ u_b^+) - \frac{1}{d_a^- + d_b^-} (d_a^- u_a^- + d_b^- u_b^-) \right] \tag{3.15}$$

3.2.2.2 Valeurs aux faces.

Soit f une face du maillage fin $T_h(\Delta)$. Si f est une face intérieure à $T_h(\Delta)$, on définit u_f la valeur en la face f par le biais de l'équation (3.13). Par définition, cette valeur permet de garantir la continuité des flux à travers f. Si f appartient à la frontière du domaine de travail Δ, on connait explicitement la valeur de u, par le biais des conditions aux limites de Dirichlet du problème (3.8). On choisit de prendre la valeur en G_f.

$$u_f = \begin{cases} \dfrac{d_f^- a^+ u_{X^+} + d_f^+ a^- u_{X^-}}{d_f^- a^+ + d_f^+ a^-} & \text{si } f = \partial k^+ \cap \partial k^-, \\ u_D(G_f) & \text{si } f \subset \partial\Delta. \end{cases} \tag{3.16}$$

3.2.3 La méthode de Volumes Finis *VFDiam*.

Décrite en premier lieu par *Coudière* et al. dans [79], la méthode de Volumes Finis *VFDiam* est utilisée par le CEA pour résoudre des problèmes de diffusion dans des milieux où siègent de fortes anisotropies ou de forts contrastes de diffusivités [20], ce qui est le cas, notamment, des matériaux cimentaires.

Contrairement à de nombreuses méthodes de Volumes Finis, dont la méthode *VF9*, la méthode *VFDiam* s'applique aussi bien en deux qu'en trois dimensions et indépendamment de la forme des éléments du maillage (triangles, tétraèdres, quadrangles ou hexaèdres). On se concentre dans cette section sur le cadre bidimensionnel, où l'on travaille sur un maillage triangulaire. la description du cas $3D$ est disponible à l'annexe §A.

3.2.3.1 Approximation du flux à travers une face.

On considère deux éléments adjacents k^+ et k^- du maillage fin $T_h(\hat{K})$, et on note $f = S_0S_1$ leur face commune. On note G_f le barycentre de f. Les inconnues, respectivement u^+ et u^-, sont situées aux barycentres G^+ et G^- des éléments. On suppose la matrice A constante sur chaque élément, de valeurs respectives A^+ et A^-. La Figure 3.3 illustre les notations employées.

La méthode *VFDiam* s'appuie sur la construction d'un volume autour de la face f, que l'on appelle le *volume* VFDiam. Il s'agit de l'union des deux demi-élementss diamant $k_f^+ = S_0S_1G^+$ et $k_f^- = S_0S_1G^-$, de volumes respectifs V^- et V^+. On introduit ensuite G_f, le barycentre de f, et les inconnues intermédiaires u_{S_i} localisées sur les noeuds S_i.

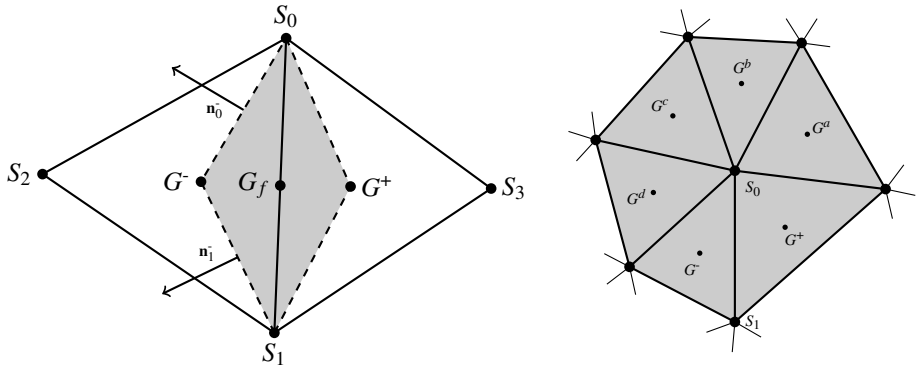


FIGURE 3.3 – À gauche : un élément *VFDiam* (en gris) est défini le long de la face commune de deux éléments adjacents. À droite : l'ensemble $\mathbb{S}_0$ (en gris) regroupe tous les éléments contenant le noeud S_O.

Afin de construire une approximation $\tilde{L}_{k^+,f}$ de $L_{k^+,f}$, on va tout d'abord moyenner la quantité $A\nabla u$ sur k_f^+ et k_f^-. Cette moyenne permet d'obtenir un premier lien entre les inconnues principales u^+ et u^-, les inconnues intermédiaires u_{S_i} et u_{G_f}, la valeur en G_f. On élimine ensuite la valeur en G_f en imposant la continuité sur f des flux moyens, sur le modèle de l'équation (3.11).

Sur le volume k_f^+, on a :

$$\begin{aligned} < A\nabla u >_{k_f^+} &= \frac{1}{V^+}\int_{k_f^+} A\nabla u \\ &= \frac{A^+}{V^+}\int_{\partial k_f^+} u\mathbf{n} \\ &= \frac{A^+}{V^+}\left(\int_{G^+S_0} u\mathbf{n} + \int_{S_0G_f} u\mathbf{n} + \int_{G_fS_1} u\mathbf{n} + \int_{S_1G^+} u\mathbf{n}\right) \end{aligned}$$

où $\mathbf{n}$ désigne le vecteur normal unitaire extérieur à k_f^+. Sur chaque face de k_f^+, on approxime u par la moyenne des valeurs aux sommets correspondants. On note de plus respectivement $\mathbf{n}_0^+$, $\mathbf{n}_1^+$ et $\mathbf{n}_f^+$ les vecteurs normaux à G^+S_0, G^+S_1 et S_0S_1, extérieurs à k_f^+, de norme $\|G^+S_0\|$, $\|G^+S_1\|$ et $\|S_0S_1\|$. On a alors :

$$< A\nabla u >_{k_f^+} \approx \frac{A^+}{V^+}\left(\frac{u^+ + u_{S_0}}{2}\mathbf{n}_0^+ + \frac{u_{S_0}+u_{G_f}}{2}\frac{\mathbf{n}_f^+}{2} + \frac{u_{G_f}+u_{S_1}}{2}\frac{\mathbf{n}_f^+}{2} + \frac{u_{S_1}+u^+}{2}\mathbf{n}_1^+\right) \tag{3.17}$$

$$\approx \frac{A^+}{2V^+}\left(-\mathbf{n}_f^+u^+ + \mathbf{n}_f^+u_{G_f} + \left(\mathbf{n}_0^+ + \frac{\mathbf{n}_f^+}{2}\right)u_{S_0} + \left(\mathbf{n}_1^+ + \frac{\mathbf{n}_f^+}{2}\right)u_{S_1}\right) \tag{3.18}$$

où on a tenu compte du fait que $\mathbf{n}_0^+ + \mathbf{n}_1^+ + \mathbf{n}_f^+ = \mathbf{0}$. De même, on peut calculer une formule similaire pour l'élément k_f^- :

$$< A\nabla u >_{k_f^-} \approx \frac{A^-}{2V^-} \left(-\mathbf{n}_f^- u^- + \mathbf{n}_f^- u_{G_f} + \left(\mathbf{n}_0^- + \frac{\mathbf{n}_f^-}{2} \right) u_{S_0} + \left(\mathbf{n}_1^- + \frac{\mathbf{n}_f^-}{2} \right) u_{S_1} \right) \quad (3.19)$$

On suppose maintenant que ces deux approximations vérifient la continuité des flux à travers la face f, c'est-à-dire que l'on a :

$$< A\nabla u >_{k_f^+} \mathbf{n}_f^+ + < A\nabla u >_{k_f^-} \mathbf{n}_f^- = 0 \quad (3.20)$$

On pose $a^+ = A^+/(2V^+)$, $\mathbf{t}_0^+ = \mathbf{n}_0^+ + \mathbf{n}_f^+/2$ et $\mathbf{t}_1^+ = \mathbf{n}_1^+ + \mathbf{n}_f^+/2$, ainsi que des quantités similaires sur k^-. On pose également $\mathbf{n}_f = \mathbf{n}_f^+ = -\mathbf{n}_f^-$ pour alléger les notations. En rapportant (3.18) et (3.19) dans (3.20), il est possible d'obtenir une expression de u_{G_f} :

$$\begin{aligned} u_{G_f} = \frac{u^+(a^+\mathbf{n}_f)\cdot\mathbf{n}_f + u^-(a^-\mathbf{n}_f)\cdot\mathbf{n}_f + u_{S_0}\left[(a^-\mathbf{t}_0^-)\cdot\mathbf{n}_f - (a^+\mathbf{t}_0^+)\cdot\mathbf{n}_f\right]}{(a^+\mathbf{n}_f)\cdot\mathbf{n}_f + (a^-\mathbf{n}_f)\cdot\mathbf{n}_f} \\ + u_{S_1}\frac{(a^-\mathbf{t}_1^-)\cdot\mathbf{n}_f - (a^+\mathbf{t}_1^+)\cdot\mathbf{n}_f}{(a^+\mathbf{n}_f)\cdot\mathbf{n}_f + (a^-\mathbf{n}_f)\cdot\mathbf{n}_f} \end{aligned} \quad (3.21)$$

En reportant (3.21) dans (3.18) ou (3.19), on dispose de l'approximation de flux suivante :

$$\begin{aligned} \tilde{L}_{k^+,f} &= < A\nabla u >_{k_f^+} \cdot \mathbf{n}_f^+ \\ \tilde{L}_{k^+,f} &= \frac{(a^-\mathbf{n}_f)\cdot\mathbf{n}_f (a^+\mathbf{n}_f)\cdot\mathbf{n}_f}{(a^+\mathbf{n}_f)\cdot\mathbf{n}_f + (a^-\mathbf{n}_f)\cdot\mathbf{n}_f}(u^- - u^+) \\ &+ \frac{(a^-\mathbf{n}_f)\cdot\mathbf{n}_f (a^+\mathbf{t}_0^+)\cdot\mathbf{n}_f + (a^+\mathbf{n}_f)\cdot\mathbf{n}_f (a^-\mathbf{t}_0^-)\cdot\mathbf{n}_f}{(a^+\mathbf{n}_f)\cdot\mathbf{n}_f + (a^-\mathbf{n}_f)\cdot\mathbf{n}_f} u_{S_0} \\ &+ \frac{(a^-\mathbf{n}_f)\cdot\mathbf{n}_f (a^+\mathbf{t}_1^+)\cdot\mathbf{n}_f + (a^+\mathbf{n}_f)\cdot\mathbf{n}_f (a^-\mathbf{t}_1^-)\cdot\mathbf{n}_f}{(a^+\mathbf{n}_f)\cdot\mathbf{n}_f + (a^-\mathbf{n}_f)\cdot\mathbf{n}_f} u_{S_1} \end{aligned} \quad (3.22)$$

Pour obtenir cette approximation, on a introduit les inconnues intermédiaires $\{u_{S_i}\}$. Pour que le problème reste bien posé, il devient donc nécessaire d'imposer une relation supplémentaire liant nos inconnues. On utilise pour cela une méthode d'interpolation par moindres carrés sur $\mathbb{S}_i$, l'ensemble des éléments incidents au noeud S_i (cf. Figure 3.3, partie droite). On suppose que u est linéaire sur $\mathbb{S}_i$, et qu'il existe donc des coefficients $(w_{i,j})_{0\le j\le d}$ tels que :

$$u(\mathbf{x}) = w_{i,0} + \sum_{1\le j\le d} w_{i,j} x_j \qquad \text{sur } \mathbb{S}_i \quad (3.23)$$

En appliquant (3.23) aux barycentres $\{G_j\}$ des éléments composants $\mathbb{S}_i$, on obtient un système linéaire liant les coefficients $(w_{i,j})_{0\leq j\leq d}$ aux inconnues principales $\{u^j\}$. Ce système est ensuite résolu par décomposition en valeurs singulières [19, 40, 61]. Par le biais de (3.23), u_{S_i} est alors exprimé comme combinaison linéaire des inconnues principales $\{u^j\}$.

3.2.3.2 Valeurs aux faces.

Afin de déterminer une valeur approchée de u en la face f, on utilise les inconnues intermédiaires $\{u_{S_i}\}$ introduites par le schéma *VFDiam*. Une fois le problème (3.8) résolu, il est facile de calculer ces valeurs à partir des inconnues principales $\{u_i\}$. On calcule alors sur la face $f = \{S_i\}$ intérieure au maillage fin, la moyenne des valeurs $\{u_{S_i}\}$ correspondantes. C'est cette valeur que l'on choisit comme valeur de $\tilde{u}_f$.

Si f appartient à $\partial\Delta$, la frontière du domaine Δ, on connait explicitement la valeur de u par le biais des conditions aux limites de Dirichlet du problème (3.8). On choisit de prendre la valeur en G_f.

On a donc :

$$u_f = \begin{cases} u_D(G_f) & \text{si } f \subset \partial\Delta. \\ \dfrac{1}{n_f}\displaystyle\sum_{S_i\in f} u_{S_i} & \text{sinon} \end{cases} \tag{3.24}$$

où n_f est le nombre de sommets de la face f.

3.3 Résolution du problème grossier.

Au cours de ces travaux, deux schémas grossiers ont été étudiés.

Tout d'abord, le problème grossier a été résolu par une méthode d'Éléments Finis. On a choisi de garder la méthode conforme, c'est-à-dire de ne pas tenir compte des discontinuités créées par la méthode de sur-échantillonnage. En effet, les précédents usages en la matière donnaient à penser que ces discontinuités resteraient faibles et ne pénaliseraient pas la méthode [90]. Comme on le verra au chapitre §4, cela n'a pas été le cas pour les exemples d'applications de ces travaux de thèse, où les contraintes de diffusivités sont très fortes.

C'est pourquoi on a implémenté une seconde méthode grossière, qui prend cette fois-ci en compte la non-conformité de l'espace discret grossier V_H. Le choix s'est porté sur une méthode discontinue de Galerkin à pénalité intérieure [29].

On rappelle que Ω est un domaine rectangulaire, respectivement parallélépipédique, et $T_H(\Omega)$ sa discrétisation grossière en rectangles, respectivement parallélépipèdes, issue de la définition 3.1.

On suppose en outre que le maillage $T_H(\Omega)$ est cohérent avec la partition de la frontière $\partial\Omega$ en (Γ_D, Γ_N), les surfaces où s'appliquent respectivement les condi-

tions aux limites de Dirichlet et de Neumann du problème (3.1). À partir des faces de $T_H(\Omega)$, on peut donc définir $T_H(\Gamma_D)$ et $T_H(\Gamma_N)$ les discrétisations de Γ_D et Γ_N.

3.3.1 Méthode des Éléments Finis Q_1.

La première méthode grossière que l'on a testé fut une méthode d'Éléments Finis Q_1 [50, 94]. Elle a tout d'abord été implémentée dans une maquette Python indépendante (§4) puis intégrée au sein du code de calcul *MPCube* (§5).

3.3.1.1 L'espace de discrétisation V_H.

Sur chaque macroélément $K \in T_H(\Omega)$, on construit les fonctions $(\Phi^i_K)_{1\leq i\leq n}$ par combinaison linéaire des fonctions $(\Psi^i_{\hat{K}})_{1\leq i\leq n}$, les solutions des problèmes de cellules (3.6). On détermine les coefficients $a_{i,l}$ en imposant, pour tout $1 \leq i \leq n$:

$$\forall 1 \leq j \leq n \qquad \Phi^i_K(S_j) = \sum_{1\leq l\leq n} a_{i,l}\Psi^l_{\hat{K}}(S_j) = \delta_{i,j}. \tag{3.25}$$

Soit $\mathscr{I}$ l'ensemble des nodes du maillage grossier $T_H(\Omega)$, et $N_{\mathscr{I}}$ le cardinal de $\mathscr{I}$. On définit la fonction Φ^I, associée au noeud $I \in \mathscr{I}$, morceau par morceau sur chaque macroélément K contenant I. Si I coincide avec le sommet S_j de K, on a donc :

$$\Phi^I_{|K} = \Phi^j_K.$$

On définit alors l'espace de discrétisation V_H par :

$$V_H = Vect((\Phi^I)_{I\in\mathscr{I}}). \tag{3.26}$$

À cause du sur-échantillonnage des cellules, les fonctions $(\Phi^I)_{I\in\mathscr{I}}$ ne sont pas nécessairement continues à travers les frontières ∂K des macro-éléments K. La méthode Q_1 est donc *non-conforme*. En pratique cependant, on suppose que cette non-conformité est négligeable, et on n'en tient pas compte dans l'implémentation de la méthode.

Il est à noter que sur-échantillonner les macroéléments n'est pas strictement nécessaire à la méthode multiéchelle. On peut choisir un découpage du milieu où $\rho = 0$ et donc avoir $\hat{K} = K$. Dans ce cas, cette étape de la méthode devient triviale, puisque l'on a :

$$\forall K, \forall 1 \leq i \leq n \qquad \Phi^i_K = \Psi^i_{\hat{K}}$$

Cependant, quand ρ est nul, la méthode multi-échelle est moins efficace. En effet, de part les conditions aux limites (3.7) imposées aux problèmes de cellules

(3.6), les fonctions $(\hat{\Psi}^l_{\hat{K}})_{1\leq l\leq n}$ sont linéaires sur les frontières $\partial\hat{K}$. Sans recouvrement, il en est alors de même des fonctions $(\Phi^I)_{I\in\mathcal{I}}$, et donc également de la solution au problème (3.2). Imposer ainsi une linéarité artificielle sur une partie du milieu pénalise la méthode en créant des *effets de bords* le long des frontières des macroéléments K. Ce problème a été identifié très tôt dans l'histoire de la méthode d'Éléments Finis multi-échelles, dès l'article originel de *Hou et Wu* [120]. Les auteurs ont alors proposé deux techniques pour résoudre ce problème : utiliser des conditions limites oscillantes (cf. §3.2.1) ou une méthode du sur-échantillonnage. Les apports qualitatifs et quantitatifs du sur-échantillonnage ρ n'ont cependant été démontrés que plus tard [90].

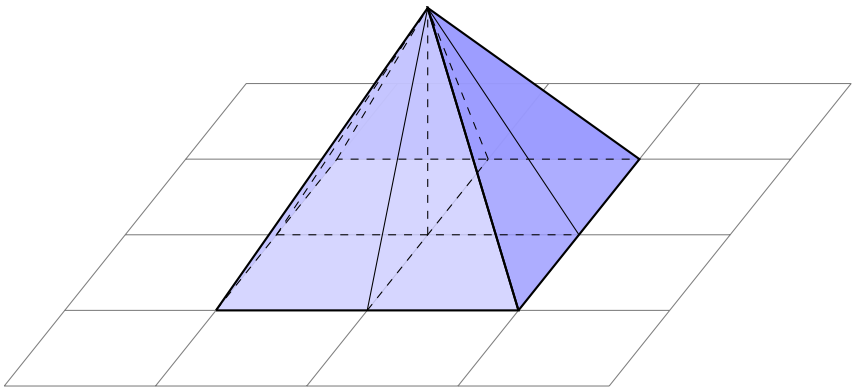


FIGURE 3.4 – Méthode d'Éléments Finis : un élément Φ^I de la base de V_H a pour support tous les macroélément $K \in T_H(\Omega)$ contenant le noeud I.

3.3.1.2 Problème discret.

Pour tout $v_H \in V_H$, $v_H = 0$ sur Γ_D, on peut écrire (3.1) sous la forme intégrale suivante :

$$\begin{aligned} \int_\Omega D\nabla C\cdot\nabla v_H &= \int_\Omega f v_H + \int_{\Gamma_N} g_N v_h \\ C &= g_D \qquad\qquad \text{sur } \Gamma_D \end{aligned}$$

On note $T_H(\Gamma_D)$ et $T_H(\Gamma_N)$ les discrétisations respectives de Γ_D et Γ_N. Le problème grossier (3.2) est alors défini par les fonctions B_H et a_H suivantes :

$$\forall (v_H, w_H) \in V_H^2 \qquad B_H(u_H, v_H) = \int_\Omega D\nabla u_H\cdot\nabla v_H \tag{3.27}$$

$$\forall v_H \in V_H \qquad a_H(v_H) = \int_\Omega f v_H + \int_{\Gamma_N} g_N v_H \tag{3.28}$$

$$\text{Sur } \Gamma_D \qquad C_H = g_D. \tag{3.29}$$

3.3.1.3 Forme matricielle.

Soient $\mathscr{I}$, $\mathscr{I}_N$ et $\mathscr{I}_D$ les ensembles des noeuds du maillage grossier $T_H(\Omega)$ décrivant respectivement Ω, Γ_N et Γ_D. On suppose les termes source, de Neumann et de Dirichlet suffisamment réguliers, et on les assimile alors à leur projection sur V_H.

$$f = \sum_{I \in \mathscr{I}} F(I)\Phi_I \tag{3.30}$$

$$g_D = \sum_{I \in \mathscr{I}_D} G_D(I)\Phi_I \tag{3.31}$$

$$g_N = \sum_{I \in \mathscr{I}_N} G_N(I)\Phi_I. \tag{3.32}$$

Résoudre le problème discrétisé (3.2) revient alors à résoudre le problème matriciel suivant :

$$\mathscr{L}_D(\mathbb{K})C_H = \mathscr{S}_D\left(\mathbb{M}F + \mathbb{M}_b G_N\right), \tag{3.33}$$

où $\mathscr{L}_D$ et $\mathscr{S}_D$ représentent un traitement particulier permettant d'imposer les conditions aux limites de Dirichlet au système linéaire. Il est décrit plus bas, à la section §3.3.1.4. Les matrices de *raideur* $\mathbb{K}$, de *masse* $\mathbb{M}$ et de *masse de bord* $\mathbb{M}_b$ sont définies par :

$$\forall (I,J) \in \mathscr{I}^2 \qquad \mathbb{K}(I,J) = \int_\Omega D\nabla\Phi^I \cdot \nabla\Phi^J. \tag{3.34}$$

$$\forall (I,J) \in \mathscr{I}^2 \qquad \mathbb{M}(I,J) = \int_\Omega \Phi^I\Phi^J. \tag{3.35}$$

$$\forall (I,J) \in \mathscr{I}_N \qquad \mathbb{M}_b(I,J) = \int_{\Gamma_N} \Phi^I\Phi^J. \tag{3.36}$$

Ces trois matrices sont assemblées à partir de leurs versions locales $\mathbb{K}_K$, $\mathbb{M}_K$ et $\mathbb{M}_{b,F}$, où $K \in T_H(\Omega)$ et $F \in T_H(\Gamma_N)$. Cela permet d'une part de distribuer le calcul fin sur plusieurs processeurs, et d'autre part de s'affranchir des sauts aux interfaces ∂K que peuvent présenter les fonctions $(\Phi^I)_{I\in\mathscr{I}}$. De cette manière, on n'utilise qu'une seule méthode que les Éléments Finis soient conformes ($\rho = 0$) ou non ($\rho \neq 0$), méthode dont la stabilité est prouvée [169].

$$\forall 1 \leq i,j \leq n \qquad \mathbb{K}_K(i,j) = \int_K D\nabla\Phi_K^i \cdot \nabla\Phi_K^j \tag{3.37}$$

Matrice de raideur $\mathbb{K}$. On calcule, à l'échelle fine, les versions locales $\mathbb{K}_K$ de la matrice de raideur.

Pour tout macroélément $K \in T_H(\Omega)$, les fonctions $(\Phi_K^i)_{1\leq i\leq n}$ sont combinaisons linéaires des $(\Psi_{\hat{K}}^i)_{1\leq i\leq n}$, les solutions des problèmes de cellules (3.6). Sur K, elles vérifient donc :

$$-\nabla\cdot(D\nabla\Phi_K^i) = 0.$$

En utilisant la formule de Green sur (3.37), et en supposant D symétrique, il vient :

$$\begin{aligned}\forall 1\leq i,j\leq n \quad \mathbb{K}_K(i,j) &= \frac{1}{2}\left(\int_{\partial K}\Phi_K^i(D\nabla\Phi_K^j\cdot\mathbf{n}) + \int_{\partial K}\Phi_K^j(D\nabla\Phi_K^i\cdot\mathbf{n})\right)\\ &\approx \frac{1}{2}\sum_{f\in\partial K}\left(\Phi_K^i(f)\int_f(D\nabla\Phi_K^j\cdot\mathbf{n}) + \Phi_K^j(f)\int_f(D\nabla\Phi_K^i\cdot\mathbf{n})\right)\end{aligned} \tag{3.38}$$

où $\mathbf{n}$ est le vecteur normal sortant de ∂K. $\Phi_K^i(f)$ est une valeur interpolée de Φ_K^i en l'arête f d'un microélément de K. Ces interpolations dépendent de la méthode Volumes Finis utilisée pour résoudre les problèmes de cellules. Elle ont été définies à la section §3.2 par les équations (3.16) pour la méthode *VF9* et (3.24) pour la méthode *VFDiam*. De la même façon, les flux des $(\Phi_K^i)_{1\leq i\leq n}$ sur les faces f sont calculés grâce à l'approximation des flux réalisée par les méthodes des Volumes Finis, voir l'équation (3.15) pour la méthode *VF9* et l'équation (3.22) pour la méthode *VFDiam*.

Matrice de masse $\mathbb{M}$. $\mathbb{M}_K$ est construit à partir des valeurs, à l'échelle fine, des fonctions $(\Phi_K^i)_{1\leq i\leq n}$. On a :

$$\begin{aligned}\forall 1\leq i,j\leq n \qquad \mathbb{M}_K(i,j) &= \int_K\Phi_K^i\Phi_K^j\\ &\approx \sum_{k\in K}|k|\Phi_K^i(k)\Phi_K^j(k)\end{aligned} \tag{3.39}$$

Matrice de masse de bord $\mathbb{M}_b$. Le calcul de $\mathbb{M}_b$ se décompose très simplement en un ensemble de calculs locaux sur les faces $F \in T_H(\Gamma_N)$. Compte tenu des supports des fonctions $(\Phi^I)_{I\in\mathscr{I}}$, on a :

$$\int_F\Phi^I\Phi^J = \begin{cases} 0 & \textit{si } I\notin F \textit{ ou } J\notin F\\ \int_F\Phi_K^{\chi(K,I)}\Phi_K^{\chi(K,J)} & \textit{sinon}\end{cases} \tag{3.40}$$

où K est l'unique macro-élément vérifiant $F\in\partial K$, et $\chi(K,I) = j$ si le noeud I coïncide avec le sommet S_j de K. Cette nouvelle intégrale est alors approximée, au niveau fin, de la manière suivante :

$$\int_F \Phi_K^i \Phi_K^j \approx \sum_{f \in F} |f| \Phi_K^i(f) \Phi_K^j(f). \tag{3.41}$$

3.3.1.4 Construction du second membre : $\mathscr{L}_D$ et $\mathscr{S}_D$.

On impose maintenant les conditions aux limites de Dirichlet au problème discrétisé. Pour cela, on élimine du problème linéaire, d'une certaine façon, les noeuds $\mathscr{I}_D$ de $\Gamma_{D,H}$. On s'affranchit ainsi du besoin de calculer un relèvement $\tilde{g}_D$ de g_D sur le domaine Ω.

Ce traitement calcule également $\mathscr{L}_D(\mathbb{K})$ à partir de $\mathbb{K}$, rendant le problème linéaire bien posé. En effet, la matrice $\mathbb{K}$ élaborée directement à partir des matrices locales $\mathbb{K}_k$ n'est pas inversible.

On pose $T = \mathbb{M}F + \mathbb{M}_b G_N$. La construction de $\mathscr{L}_D(\mathbb{K})$ et $\mathscr{S}_D(T)$ suit alors l'Algorithme 3.2, extrait de [50] :

Algorithme 3.2 Application des conditions aux limites de Dirichlet.

$\mathscr{L}_D(\mathbb{K}) \leftarrow \mathbb{K}$.
$\mathscr{S}_D(T) \leftarrow T - \mathbb{K}G_D$.
Pour tout $I \in \mathscr{I}_D$ **faire**
    $\mathscr{S}_D(T)(I) \leftarrow \mathbb{K}(I,I)G_D(I)$
    Pour tout $I \in \mathscr{I} \setminus \{I\}$ **faire**
        $\mathscr{L}_D(\mathbb{K})(I,J) \leftarrow 0$
        $\mathscr{L}_D(\mathbb{K})(J,I) \leftarrow 0$
    Fin Pour
Fin Pour
Renvoi $\mathscr{L}_D(\mathbb{K})$ et $\mathscr{S}_D(T)$.

3.3.2 Méthode discontinue de Galerkin.

3.3.2.1 Choix de la méthode.

La première méthode discontinue de Galerkin a été introduite par *Reed et Hill* [158] dans le cadre des équations hyperboliques. Durant la même période, des méthodes de Galerkin, utilisant des Éléments Finis discontinus, ont été développées dans le cadre des équations paraboliques et elliptiques, par exemple par *Babuška et Zlámal* [31]. Les méthodes de Galerkin discontinues ont depuis été utilisées sur un grand nombre d'équations différentielles. Pour un état de l'art sur ce sujet, on pourra se référer aux travaux de *Cockurn* [74] ou à ceux de *Ern et Guermond* [94].

En 2002, *Arnold* et al. [30] ont réalisé une analyse unifiée des méthodes discontinues de Galerkin pour les problèmes elliptiques, reprenant les méthodes de *Babuška et Zlámal* [31], *Arnold* (méthode *Interior Penalty* ou IP) [29, 82], *Bassi et Rebay* [37], *Bassi* et al. [38], *Cockburn et Shu* [75] (méthode *Local Discontinuous*

Galerkin ou LDG) , *Baumann et Oden* [39], *Rivière* et al. [160] (méthode *Non Symetric Interior Penalty Galerkin* ou NIPG) et *Brezzi* et al. [56, 57]. Ce travail a ensuite été complété par *Castillo* [64] qui a évalué les performances numériques de ces méthodes en termes de stabilité, taux de convergence, de précision et de coût stockage.

On s'est concentrés ici sur les méthodes symétriques, comme IP et LDG, qui apparaissent, dans cette étude, plus stables. Leur taux de convergence, notamment, est peu volatile aux fluctuations du terme de pénalisation. À choisir entre les méthodes LDG et IP, c'est cette dernière qui l'a emporté, pour plusieurs raisons.

Tout d'abord, le seul avantage de la méthode LDG vis-à-vis de la méthode IP semble être sa plus grande stabilité. En effet, le terme de pénalisation LDG ne dépend ni du maillage grossier ni de l'ordre des polynômes utilisés. Le schéma grossier étant particulièrement simple (éléments quadrangulaires réguliers, polynomes d'ordre 1), il est peu probable que cette stabilité soit réellement utile.

De plus, la méthode IP ne nécessite qu'un *stencil* peu étendu, chaque macroélément n'interagissant qu'avec ses voisins directs. La méthode LDG, au contraire, à un stencil bien plus grand, puisque chaque macroélément interagit avec ses voisins et les voisins de ceux-ci. Le calcul des termes matriciels est l'étape la plus complexe à mettre en œuvre dans les méthodes multi-échelles. il est donc intéressant de les limiter autant que possible.

3.3.2.2 Problème discret.

On considère les fonctions $(\Phi_K^i)_{1\leq i\leq n}$ définies par (3.25) sur chaque macroélément $K \in T_H(\Omega)$. On les renumérote par macroélément et sommet i croissant, en l'ensemble de fonctions $(\Phi^I)_{I\in\mathscr{I}}$. On appelle V_H l'espace vectoriel engendré par les fonctions $(\Phi^I)_{I\in\mathscr{I}}$, qui est donc de dimension $2^d \times Card(T_H(\Omega))$. La Figure 3.5 présente un exemple de fonction Φ^I. Dans tout ce qui suit, soit $I \in \mathscr{I}$, on note $K(I)$ et $s(I)$, le macroélément et le sommet tel que :

$$\Phi^I = \Phi^{s(I)}_{K(I)} \tag{3.42}$$

On rappelle que si $T_H(\Omega)$ est la discrétisation du milieu Ω, on note $T_H(\Gamma_D)$ et $T_H(\Gamma_N)$ les discrétisations respectives de Γ_D et Γ_N. On note Γ_H^0 l'ensemble des faces intérieures à la discrétisation, c'est-à-dire commune à deux éléments de $T_H(\Omega)$.

Pour tout $v_H \in V_H$, en utilisant la formule de Green sur chaque macroélément K de $T_H(\Omega)$, on peut écrire (3.1) sous la forme intégrale suivante :

$$\sum_{K\in T_H(\Omega)} \left(\int_K D\nabla C\cdot\nabla v_H - \sum_{F\in\partial K} \int_F D\nabla C\cdot v_H \mathbf{n}_{K,F} \right) = \sum_{K\in T_H(\Omega)} \int_K f v_H \tag{3.43}$$

où $\mathbf{n}_{K,F}$ est le vecteur normal à la face F sortant de K.

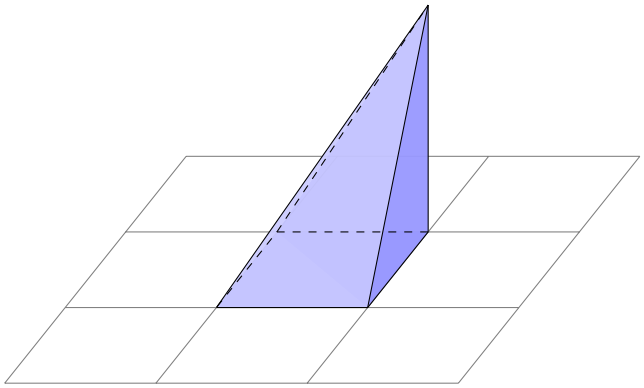


FIGURE 3.5 – Méthode Discontinue de Galerkin : un élément Φ^I de la base de V_H a pour support un unique macroélément $K \in T_H(\Omega)$.

Définition 3.5 *Soit $F \in \Gamma_H^0$ l'arête commune aux deux éléments K_1 et K_2 de $T_H(\Omega)$. Soit ζ une fontion continue par éléments, dont on note ζ_1 et ζ_2 la trace sur F suivant les éléments K_1 et K_2. On définit le* saut $[\![\zeta]\!]$ *et la* moyenne $\{\zeta\}$ *d'une quantité ζ à travers F par :*

$$[\![\zeta]\!] = \zeta_1 \mathbf{n}_{K_1,F} + \zeta_2 \mathbf{n}_{K_2,F} \qquad \{\zeta\} = \frac{1}{2}(\zeta_1 + \zeta_2).$$

Il est à noter que $[\![\zeta]\!]$ est un vecteur si ζ est une valeur scalaire et vice-versa. On étend ces définitions au cas des faces $F \in K$ de $T_H(\Gamma_D)$ en posant :

$$[\![\zeta]\!] = \zeta \mathbf{n}_{K,F} \qquad \{\zeta\} = \zeta.$$

En tenant compte de la condition limite de Neumann, on peut alors réécrire (3.43) sous la forme :

$$\sum_{K \in T_H(\Omega)} \int_K D\nabla C \cdot \nabla v_H - \sum_{F \in \Gamma_H^0 \cup T_H(\Gamma_D)} \int_F \{D\nabla C\} \cdot [\![v_H]\!] = \sum_{K \in T_H(\Omega)} \int_K f v_H + \sum_{F \in T_H(\Gamma_N)} \int_F g_N v_H. \quad (3.44)$$

Pour rendre symétrique cette équation, on soustrait un terme $\int_F \{D\nabla v_H\} \cdot [\![C]\!]$, qui est nul sur toutes les faces de Γ_H^0, puisque C est continue. En tenant compte de la condition limite de Dirichlet, on a donc :

$$\sum_{K\in T_H(\Omega)}\int_K D\nabla C\cdot\nabla v_H - \sum_{F\in\Gamma_H^0\cup T_H(\Gamma_D)}\int_F (\{D\nabla C\}\cdot[\![v_H]\!]+\{D\nabla v_H\}\cdot[\![C]\!])$$
$$= \sum_{K\in T_H(\Omega)}\int_K f v_H + \sum_{F\in T_H(\Gamma_N)}\int_F g_N v_H$$
$$- \sum_{F\in T_H(\Gamma_D)}\int_F g_D(D\nabla v_H)\cdot\mathbf{n}_F.$$

Enfin, on ajoute un terme de pénalisation sur chaque face F qui impose faiblement la continuité à notre solution approchée C_H. De la forme $\int_F \mu_F[\![C]\!]\cdot[\![v_H]\!]$, il est, comme le terme de symétrie, nul sur toutes les faces de Γ_H^0 car C est continue. On pose $\mu_F = \mu h_F^{-1}$, où $\mu > 0$ est un terme de pénalisation indépendant de la taille du maillage et h_F le diamètre de la face F, et on a alors :

$$\sum_{K\in T_H(\Omega)}\int_K D\nabla C\cdot\nabla v_H - \sum_{F\in\Gamma_H^0\cup T_H(\Gamma_D)}\int_F (\{D\nabla C\}\cdot[\![v_H]\!]+\{D\nabla v_H\}\cdot[\![C]\!])$$
$$+ \sum_{F\in\Gamma_H^0\cup T_H(\Gamma_D)}\int_F \mu_F[\![C]\!]\cdot[\![v_H]\!]$$
$$= \sum_{K\in T_H(\Omega)}\int_K f v_H + \sum_{F\in T_H(\Gamma_N)}\int_F g_N v_H$$
$$- \sum_{F\in T_H(\Gamma_D)}\int_F g_D(D\nabla v_H)\cdot\mathbf{n}_F + \sum_{F\in T_H(\Gamma_D)}\int_F \mu_F[\![g_D]\!]\cdot[\![v_H]\!].$$

Finalement, le problème grossier (3.2) est défini par les fonctions B_H et a_H suivantes :

$$\forall (u_H,v_H)\in V_H^2 \quad B_H(u_H,v_H) = \sum_{K\in T_H(\Omega)}\int_K D\nabla u_H\cdot\nabla v_H$$
$$- \sum_{F\in\Gamma_H^0\cup T_H(\Gamma_D)}\int_F (\{D\nabla u_H\}\cdot[\![v_H]\!]+\{D\nabla v_H\}\cdot[\![u_H]\!])$$
$$+ \sum_{F\in\Gamma_H^0\cup T_H(\Gamma_D)}\int_F \mu_F[\![u_H]\!]\cdot[\![v_H]\!]$$
$$\forall v_H\in V_H \qquad a_H(v_H) = \sum_{K\in T_H(\Omega)}\int_K f v_H + \sum_{F\in T_H(\Gamma_N)}\int_F g_N v_H$$
$$- \sum_{F\in T_H(\Gamma_D)}\int_F g_D(D\nabla v_H)\cdot\mathbf{n}_F$$
$$+ \sum_{F\in T_H(\Gamma_D)}\int_F \mu_F[\![g_D]\!]\cdot[\![v_H]\!]. \tag{3.45}$$

3.3.2.3 Forme matricielle.

Comme pour la méthode des Éléments Finis, on suppose les termes source, de Neumann et de Dirichlet suffisamment réguliers, et on les assimile alors à leur projection sur V_H. Les formules de projection (3.30), (3.31) et (3.32) sont donc valides.

Résoudre le problème discrétisé (3.2) revient alors à résoudre le problème matriciel suivant :

$$(\mathbb{K}-\mathbb{T}+\mathbb{Y})C_H = \mathbb{M}F + \mathbb{M}_b G_N - \mathbb{T}_b G_D + \mathbb{Y}_b G_D \tag{3.46}$$

où les matrices de raideur $\mathbb{K}$, de masse $\mathbb{M}$ et de masse de bord $\mathbb{M}_b$ ont été précédemment définies par les formules (3.34), (3.35) et (3.36). Les matrices de *saut* $\mathbb{T}$, de *pénalisation* $\mathbb{Y}$, de *saut Dirichlet* $\mathbb{T}_b$ et de *pénalisation Dirichlet* $\mathbb{Y}_b$ sont quant à elles définies par :

$$\forall (I,J)\in\mathscr{I}^2 \qquad \mathbb{T}(I,J) = \sum_{F\in\Gamma_H^0\cup T_H(\Gamma_D)} \int_F \left(\left\{D\nabla\Phi^I\right\}\cdot[\![\Phi^J]\!] + \left\{D\nabla\Phi^J\right\}\cdot[\![\Phi^I]\!]\right), \tag{3.47}$$

$$\forall (I,J)\in\mathscr{I}^2 \qquad \mathbb{Y}(I,J) = \sum_{F\in\Gamma_H^0\cup T_H(\Gamma_D)} \int_F \mu_F [\![\Phi^I]\!]\cdot[\![\Phi^J]\!], \tag{3.48}$$

$$\forall I\in\mathscr{I},\forall J\in\mathscr{I}_D \qquad \mathbb{T}_b(I,J) = \sum_{F\in T_H(\Gamma_D)} \int_F (D\nabla\Phi^J\cdot\mathbf{n}_F)\Phi^J, \tag{3.49}$$

$$\forall I\in\mathscr{I},\forall J\in\mathscr{I}_D \qquad \mathbb{Y}_b(I,J) = \sum_{F\in T_H(\Gamma_D)} \int_F \mu_F \Phi^I\Phi^J. \tag{3.50}$$

Les matrices $\mathbb{K}$, $\mathbb{M}$ et $\mathbb{M}_b$ sont assemblées à partir de leurs versions locales, que l'on a calculées sur chaque $K\in T_H(\Omega)$. L'assemblage est légèrement différent du cas des Éléments Finis ; la numérotation des noeuds n'étant pas la même ; mais respecte le même principe. On a :

$$\forall (I,J)\in\mathscr{I}^2 \qquad \mathbb{K}(I,J) = \delta_{K(I),K(J)} \int_{K(I)} D\nabla\Phi^{s(I)}_{K(I)}\cdot\nabla\Phi^{s(J)}_{K(J)}. \tag{3.51}$$

Une formule similaire existe pour $\mathbb{M}$ et $\mathbb{M}_b$.

Les autres matrices intervenant dans le problème matriciel (3.46) nécessitent de calculer des termes de saut et de moyenne à travers les arêtes des macroéléments. Compte tenu des supports des fonctions $(\Phi^I)_{I\in\mathscr{I}}$, bon nombre des termes de saut et des moyenne sont nuls. Soit F une face de $\Gamma_H^0\cup T_H(\Gamma_D)$, pour tout $(I,J)\in\mathscr{I}^2$, on pose :

$$\varepsilon(I,J,F) = \begin{cases} 1 & \text{si } K(I) = K(J), \\ -1 & \text{si } \partial K(I) \cap \partial K(J) = F, \\ 0 & \text{sinon,} \end{cases}$$

On a alors :

$$\forall (I,J) \in \mathscr{I}^2 \qquad \int_F \mu_F [\![\Phi^I]\!] \cdot [\![\Phi^J]\!] = \varepsilon(I,J,F) \int_F \mu_F \Phi^{s(I)}_{K(I)} \Phi^{s(J)}_{K(J)},$$

les autres termes des matrices de saut et de pénalité se réécrivant de façon identique.

3.3.2.4 Calcul des termes matriciels.

Soit F une face de $\Gamma^0_H \cup T_H(\Gamma_D)$, on suppose disposer d'une discrétisation fine $T_h(F)$ de F sur laquelle des valeurs de Φ^I et de $D\nabla\Phi^I$ ont été interpolées des deux côtés K_1 et K_2. En pratique, cela suppose que les maillages fins $T_h(K_1)$ de K_1 et $T_h(K_2)$ de K_2 coïncident sur F. On approxime alors les termes de saut et de pénalisation de la manière suivante :

$$\int_F \mu_F [\![\Phi^I]\!] \cdot [\![\Phi^J]\!] \approx \varepsilon(I,J,F) \mu_F \sum_{f \in T_h(F)} |f| \Phi^{s(I)}_{K(I)}(f) \Phi^{s(J)}_{K(J)}(f)$$

Remarque 3.3 *Le calcul des termes des matrices de sauts et de pénalisation est représentatif des avantages et inconvénients de la méthode discontinue de Galerkin. En effet c'est l'existence de ces termes qui permet de controler plus efficacement les discontinuités créées par la méthode de sur-échantillonnage. En contrepartie, ils demandent d'imposer des contraintes fortes sur les maillages des macro-éléments, afin que ceux-ci coïncident sur les faces de calculs. Les moyens techniques mis en œuvre pour obtenir un tel résultat sont décrits à la section §6.3.*

Remarque 3.4 *Compte tenu de la forme des matrices de saut et de pénalisation, il n'est pas possible de calculer les termes macroélément par macroélement, puis d'assembler les résultats, comme on le fait pour la matrice de rigidité. On peut cependant utiliser un procédé similaire qui fonctionne face par face : les calculs de saut sont en effet techniquement indépendants et on pourrait ainsi distribuer le travail sur plusieurs processeurs.*

Cependant, pour des raisons pratiques, cette idée n'a pas été implémentée car elle est difficilement applicable au contexte de simulation MPCube. *En effet, de tels calculs nécessitent de lancer une nouveau jeu de simulations, non plus macroélément par macroélément, mais face par face. Bon nombre de valeurs, calculées au*

cours des premières simulations mais non stockées, seraient perdues d'une simulation à l'autre, demandant ainsi au programme d'effectuer un travail redondant. Il nous a donc paru préférable de calculer, macroélément par macroélément, les valeurs de chaque fonction élémentaire sur les bords de chaque macroélément, puis de stocker ces valeurs. Le calcul des matrices de saut et de pénalisation se fait donc séquentiellement lors de la résolution du problème grossier.

3.4 Reconstruction de la solution et du flux au niveau fin.

La dernière étape de l'algorithme calcule, au niveau fin, la solution $C_{H,h}$, en pondérant les fonctions $(\Phi^I)_{I\in\mathscr{I}}$ par les valeurs de la solution grossière.

$$C_{H,h} = \sum_{I\in\mathscr{I}} \mathbf{C}_H(I)\Phi^I \ . \tag{3.52}$$

Par principe, on cherche à résoudre des problèmes de *très grandes tailles* pour lesquels une résolution classique est techniquement irréalisable, soit parce que le système prend trop de temps à résoudre, soit parce que la place mémoire exigée pour la résolution et le stockage de la solution est trop importante. Il est donc généralement impossible de stocker et, à plus fortes raisons, de visualiser la concentration (ou le gradient, la densité de flux $\|D\nabla C\|$) d'un seul tenant. Par contre, il peut être intéressant de reconstruire la solution sur une zone déterminée, par exemple autour d'une singularité. On ne manipule ainsi que des blocs de tailles raisonnables, qui n'excèdent pas les capacités informatiques disponibles.

Pour reconstruire la solution sur une zone, il est nécessaire de fusionner les champs solutions sur des macroéléments adjacents. Cette tâche est plus élégante quand les maillages concernés coïncident le long des frontières des macroéléments mais, à ce stade de la méthode, ce n'est pas une obligation. En effet, deux outils seulement demandent une solution fine : la visualisation et les calculs des taux d'erreur, et tout deux n'utilisent que les volumes des maillages, non les faces. Avoir deux discrétisation disparates des ∂K est donc sans réelle importance.

De plus, ni la méthode des Éléments Finis, ni la méthode de Galerkin Discontinue n'étant conformes, aucune forme de continuité n'est imposée à la solution le long des ∂K, quand bien même les maillages y coïncideraient.

3.5 Estimation du coût de calcul.

On présente maintenant une estimation du coût en temps de calcul des méthodes multi-échelles. On divise le domaine en M macroéléments, et chaque macroélément en N microéléments. Compte tenu du recouvrement, chaque cellule dispose donc d'environ $(1+2\rho)^d N$ microéléments. Soit $\mathscr{F}(\mathscr{N})$ une fonction estimant

le coût, en termes de calcul, de la résolution d'un problème linéaire de taille $\mathscr{N}$, le Tableau 3.1 reprend le coût approché des diverses parties de la méthode.

Étapes	Coût
Division du domaine	$M \times O(N)$
Problèmes de cellules	$M \times \mathscr{F}((1+2\rho)^d N)$
Calcul de la base	$M \times O(N)$
Calcul de K, M et M_b	$M \times O(N) + O(M)$
Problème grossier	$\mathscr{F}(M)$
Recalcul à l'échelle fine	$M \times O(N)$
Total	$M \times (\mathscr{F}((1+2\rho)^d N) + O(N)) + O(M) + \mathscr{F}(M)$

TABLE 3.1 – Coût calcul des méthodes multi-échelles.

Il s'agit ici d'une approximation des coûts de la méthode, et certaines étapes ont été ignorées. Par exemple, obtenir la base $(\Phi^I)_{I\in\mathscr{I}}$ nécessite la résolution de M problèmes linéaires à 2^d inconnues ; résolution dont le coût est négligeable devant celui de la sommation des $(\Psi_K^{\hat{s}})_{\hat{s}\in\hat{\mathscr{S}}}$ qui en résulte.

Le coût total d'une méthode multi-échelle peut-être donc être estimée par :

$$\mathscr{C}^{tot} = \underbrace{M \times \mathscr{F}((1+2\rho)^d N)}_{\text{Problèmes de cellule}} + \underbrace{\mathscr{F}(M)}_{\text{Problème grossier}} + M \times O(N), \tag{3.53}$$

où on utilise la notation de Landau $O(N)$ pour désigner une fonction d'ordre N à l'infini.

Ce résultat est à comparer à celui d'une résolution directe par la méthode des Éléments Finis ou des Volumes Finis, qui est de $\mathscr{F}(MN)$. Généralement, nous avons $\mathscr{F}(MN) = O(M^3N^3)$ si on utilise une méthode de résolution directe sur une matrice pleine, ou $\mathscr{F}(MN) = O(M^2N^2)$ pour une méthode itérative sur une matrice creuse. Les méthodes directes sont donc beaucoup plus coûteuse que la méthode multi-échelle.

À ce coût en temp calcul s'ajoute le coût en termes de place mémoire qui devient rédibitoire pour les grands et très grands problèmes. La méthode multi-échelle est dans ces cas beaucoup plus économique, car les problèmes résolus sont petits. Bien entendu, ce gain de calcul et de mémoire a un prix et les résultat obtenus par des méthodes multi-échelles sont, par principe, moins précis qu'une résolution directe.

D'autre part, il est bon de souligner le second avantage des méthodes multi-échelles sur une résolution directe : leur parallélisme. La plupart des étapes de ces méthodes peuvent en effet être effectuées en parallèle (les termes de la forme

$M \times ...$). Seules les étapes d'assemblages des matrices globales et la résolution du problème grossier ne peuvent être effectuées de front.

En supposant qu'on dispose de suffisament de processeurs, de l'ordre de M, on peut donc espérer un coût en temps de calcul très bas :

$$\mathscr{C}^{tot} = \mathscr{F}((1+2\rho)^d N) + \mathscr{F}(M) + O(N). \tag{3.54}$$

3.6 Modélisation des matériaux cimentaires.

3.6.1 Description générale.

Définition 3.6 *Dans cette thèse, sont dits* cimentaires *les deux ensembles de matériaux suivants :*

- *les pâtes de ciment, c'est-à-dire une poudre de ciment hydraté puis séché.*
- *les bétons et les mortiers.*

On fabrique un béton en mélangeant des granulats, généralement du sable et des gravillons, avec de la poudre de ciment et de l'eau. Ce mélange, une fois homogénéisé et séché, forme un tout solide, la poudre de ciment servant de liant. On parle de *mortier* quand les granulats utilisés sont très fins, se réduisant généralement à du sable. Ces trois processus, hydratation, brassage et séchage, n'affectent pas chimiquement les granulats en tant que tels [150]. Puisque chaque granulat conserve ses caractéristiques individuelles (forme, taille, diffusivité, résistance), il est alors possible de déterminer précisement celles-ci par des mesures industrielles indépendantes [89].

On ne peut utiliser une telle méthode pour la pâte de ciment, qui est affectée par la présence des granulats. En effet, lors de la fabrication du béton, on observe que la pâte de ciment adopte une structure particulière au voisinage des granulats. Ces structures jouent un rôle fondamental dans le comportement macroscopique du béton [166, 176]. On peut par exemple citer, dans les mortiers, la formation d'une *auréole de transition*, de très haute diffusivité, autour des grains de sables [52]. Alors que les grains de sables sont nécessairement séparés les uns des autres, les auréoles de transition correspondantes peuvent fusionner les unes les autres. Au delà d'une certaine concentration de sable, de véritables chemins de haute diffusivité apparaissent à travers le milieu, un phénomène appelé *percolation* [168] et illustré à la Figure 3.6.

Lors de l'étude d'un béton, l'échelle de travail choisie est généralement de l'ordre du centimètre, voire du millimètre. À cette échelle, on considère que la pâte de ciment est homogène. Il est alors possible de représenter le béton comme un ensemble d'*inclusions*, modélisant les gravillons et les grains de sable, enchâssés dans un milieu homogène, la pâte de ciment. Les structures particulières de la pâte de ciment sont alors considérées comme des agglomérats, et non plus comme de la pâte. Dans le cas du mortier décrit plus haut, les inclusions représentant le

sable possèdent alors un noyau (modélisant le grain de sable) entouré d'une fine couche (l'auréole de transition).

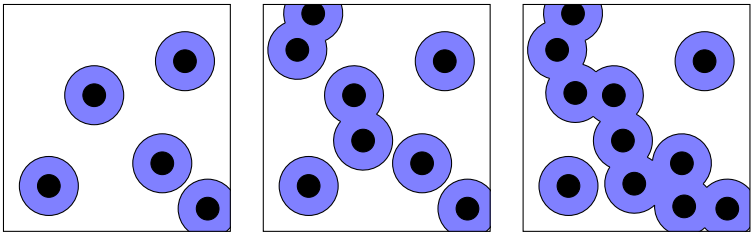


FIGURE 3.6 – Illustration du phénomène de percolation dans le béton. La pâte de ciment (en blanc) forme une auréole de transition (en bleu) autour des grains de sable (en noir). Quand le nombre de grain de sable augmente, ces auréoles fusionnent (au milieu), jusqu'à former des voies très diffusive dans le béton (à droite).

La composition de la poudre de ciment est bien connue : il s'agit d'un mélange de silice, d'alumine, de carbonate de chaux que l'on a broyé et cuit à environ $1450^{\circ}C$. Les proportions de ce mélange, ainsi que l'ajout de composants en très faible quantité, varient en fonction du ciment que l'on souhaite obtenir [148].

Pour autant, l'étude précise de la pâte de ciment en elle-même est plus ardue que celle du béton dans son ensemble. En effet, le processus permettant de passer de la poudre à la pâte de ciment (hydratation, brassage puis séchage) est le siège de nombreuses transformations chimiques [115, 150] : hydratation et précipitation des espèces chimiques, migration et réarrangement des composés en de nouvelles structures... Le résultat final, c'est-à-dire la pâte de ciment, n'a alors plus grand chose de comparable avec la poudre de ciment originelle.

Il est donc nécessaire d'analyser chaque structure de la pâte de ciment pour en déterminer les caractéristiques physico-chimiques, une analyse compliquée par l'échelle des tailles caractéristiques mises en jeu. En effet, si on se place à l'échelle microstructurale de la pâte cimentaire, la taille des microstructures minérales étudiées varie entre 1 et $30\mu m$ environ. Il est relativement aisé de différencier visuellement ces micro-structures, par exemple par microscopie optique [83] ou microscopie électronique à balayage [165]. Identifier la composition et estimer les caractérisitiques mécaniques et diffusives de chaque phase est par contre un travail de longue haleine. Généralement, une fois identifiée la composition d'une structure par spectrométrie, on recherche un minéral naturel composé de cette unique phase. On supposera alors que les caractéristiques mécaniques de cette phase dans la pâte de béton sont celles du composé «pure». On se reportera aux travaux de *Constantinides et Ulm* [78] et *Haecker* et al. [110] et aux références associées pour de plus amples détails à ce sujet.

Quatre composés principaux sont identifiables dans la pâte de ciment durcie [159]. Les *résidus anhydres* n'ont pas été chimiquement liés lors de l'hydratation

du ciment. Même s'ils sont agglomérés au sein de la pâte, ils ne participent pas au processus de diffusion. La *portlandite* est la seule phase qui se présente sous une forme relativement pure. Elle s'organise en plaquettes hexagonales. L'*ettringite* et ses dérivées, qui forment des structures en aiguilles. Enfin, les *silicates de calcium hydratés* ou $C-S-H$ forment généralement un gel aforme au sein de la pâte. Au voisinage d'autres composés, notamment les résidus anhydres, ils peuvent cependant s'organiser en couches de différentes densités (*Low Density* $C-S-H$ ou *High Density* $C-S-H$). L'espace de la pâte de ciment qui n'est pas occupé par ces structures est généralement appelé *porosité capillaire* [102].

On voit donc que ces matériaux, bien qu'agissant à des échelles différentes, possèdent en fait une structure semblable. Pour les modéliser, on place, au sein d'un milieu homogène que l'on nomme la *matrice*, une série d'*inclusions* représentant les structures importantes. Le nombre, la forme et la diffusivité associés à ces inclusions dépendent bien entendu de la phase représentée.

Dans le cas du béton, la matrice représente la pâte de ciment, tandis que les gravillons, les grains de sables et les auréoles de transitions sont représentés par des inclusions.

Dans le cas de la pâte de ciment, le $C-S-H$ compose la matrice et les anhydres, la portlandite et l'ettringite sont des inclusions. Si on veut différencier les deux types de $C-S-H$, la matrice représente la porosité capillaire, et les anhydres, la portlandite, l'ettringite, le *LD* $C-S-H$ et le *HD* $C-S-H$ sont des inclusions. Le Tableau 3.2 résume ces informations et cite quelques publications ayant utilisées une telle modélisation.

	Milieu homogène	Inclusions	Références
Béton	Pâte de ciment	Gravillons Sable Auréoles de transition	[52, 166, 176]
Pâte de ciment I	$C-S-H$	Anhydres Portlandite Ettringite	[36, 70]
Pâte de ciment II	Porosité capillaire	Anhydres Portlandite Ettringite *LD* $C-S-H$ *HD* $C-S-H$	[21, 43, 45]

TABLE 3.2 – Tableau récapitulatif de la modélisation des matériaux cimentaires sous la forme d'inclusions enchâssées dans un milieu homogène. On référence à droite quelques publications ayant utilisées ces modélisations.

3.6.2 Dégradation des matériaux cimentaires.

La dégradation des matériaux cimentaires par une solution agressive, généralement de l'eau, conduit à la mise en place de deux phénomènes physico-chimique [159]. Les gradients de concentration entre la solution interstitielle, c'est-à-dire initialement présente dans le béton, et la solution agressive initie tout d'abord un transport de matière par diffusion. Les variations de concentrations résultant de la diffusion favorisent ensuite des réactions chimiques de dissolution-précipitation.

Ces phénomènes physico-chimiques modifient les propriétés des différentes phases des matériaux cimentaires. Certaines inclusions disparaissent tout simplement du milieu, d'autres voient leurs propriétés évoluer.

Du point de vue de la simulation numérique, cette dégradation se traduit soit par un changement du champ de diffusivité, soit par une extension du domaine de travail. Le cas de l'évolution du champ de diffusivité est simple à gérer : il suffit d'effectuer une nouvelle simulation avec les valeurs adéquates. L'extension du domaine de travail demande un effort supplémentaire, au sens où il affecte la discrétisation des milieux correspondants, et non plus un simple paramètre numérique de la simulation.

Si on considère par exemple une composante non-diffusive du milieu, modélisant une espèce *anhydre*. Au sein de cette composante, il n'y a pas lieu de résoudre une équation de diffusion. Le domaine de travail tient donc compte de cette zone, et impose des frontières non-diffusives aux frontières des inclusions appartenant à cette phase. Cependant, une fois dissoute la composante, la zone correspondante est, schématiquement, vide. Elle est alors diffusive et elle joue donc un rôle dans les phénomènes de diffusion. On doit donc l'inclure dans les simulations de diffusion et d'homogénéisation.

Dans le cadre de ces travaux de thèse, on souscrit à l'hypothèse de l'équilibre chimique local, communément admise dans le cas de la dégradation des matériaux cimentaires, qui stipule que les cinétiques chimiques réactionnelles sont beaucoup plus rapides que le transport diffusif. L'étude du phénomène de dégradation prend donc ici la forme de plusieurs simulations de *diffusion stationnaire* sur des matériaux cimentaires à différents stades de dégradation. On considère de plus que les phénomènes de dégradation agissent globalement sur le milieu : il n'y a pas de *front de dégradation* [159] et les inclusions sont dégradées par jeux complets.

Au cours d'un processus de dégradation, on travaille donc sur une succession de domaines qu'il faut discrétiser. On suppose ici que cette succession de dégradations est connue à l'avance, par exemple grâce à des observations physiques. Le milieu à un instant donné ne dépend donc pas de simulations effectuées sur le domaine à des instants antérieurs comme c'est le cas entre autres pour les problèmes de *frontières mobiles* [99, 118].

Ce processus de dégradation a été mis en œuvre dans le cadre de la chaîne de calcul *SALOME-MPCube*. On présente ainsi à la section §6.4 les solutions tech-

niques permettant de générer efficacement les discrétisations associées aux étapes d'un processus de dégradation.

Deuxième partie

Mise en œuvre en dimension 2.

Chapitre 4

Implémentation de la méthode Q_1/*VF9* en dimension deux.

Introduction.

On présente dans ce chapitre une implémentation indépendante de la méthode multi-échelle, réalisée durant la première année du travail de thèse. L'objectif premier de la réalisation de cette maquette est de pouvoir prendre conscience des diverses problématiques associées à ce sujet de thèse. C'est pourquoi on s'affranchit autant que possible des contraintes techniques en imposant des hypothèses fortes au problème, pour se concentrer sur la méthode multi-échelle proprement dite.

En particulier, le domaine de travail est dans ce chapitre considéré comme un ouvert rectangulaire de $\mathbb{R}^2$ exclusivement. Il est découpé en macro-éléments K eux-aussi rectangulaires. Au niveau grossier, le problème est résolu par une méthode d'Éléments Finis Q_1, décrite à la section §3.3.1.

Au niveau fin, les problèmes de cellules sont résolus par le biais d'une méthode de Volumes Finis *VF9* [97], décrite à la section §3.2.2, sur des maillages fins quadrangulaires réguliers. Ce dernier choix pertmet de laisser de côté le problème de la discrétisation du milieu qui demande une attention particulière dans un cadre plus étendu et qui est l'objet du chapitre §6.

Le travail se fait ici séquentiellement, la parallélisation des travaux étant reportée à l'implémentation *MPCube* décrite au chapitre §5.

Le chapitre s'articule autour de deux axes. On présente tout d'abord l'implémentation des différentes étapes de la méthode, ainsi que sa validation par le biais d'un ou plusieurs exemples. On utilise ensuite cette maquette pour réaliser une simulation de diffusion au sein d'un cas pratique de matériaux cimentaires. Les résultats présentés dans ce chapitre ont été partiellement rédigés dans un rapport CEA [9].

4.1 Implémentation.

L'implémentation de la méthode Q_1/*VF9* a été réalisée en language Python (version 2.4 et supérieure) [155]. Comme on l'a décrit dans le chapitre précédent, la méthode multi-échelle se découpe en plusieurs étapes (génération du milieu, résolution des problèmes de cellules, assemblage de la base d'Éléments Finis, etc.), chaque étape utilisant les résultats de la précédente.

Ce type de chaîne de calcul peut s'avérer très vulnérable aux erreurs, surtout si le travail est fait *à la volée*, c'est-à-dire sans stockage de résultats temporaires. En effet, si on ne stocke que les résultats finaux, i.e la solution reconstruite fine $C_{H,h}$, une erreur intervant à la dernière étape de la chaîne de calcul entraînera la perte de toutes les données précédemment calculées. Il sera alors nécessaire de reprendre le travail sur la chaîne de calcul depuis le début.

De plus, travailler *à la volée* réduit considérablement l'intérêt de la méthode multi-échelle. En effet, les problèmes de cellules et le problème grossier sont découplés : une fois les problèmes de cellules résolus, n'importe quel problème grossier peut être résolu. Il est donc vital de stocker les solutions des problèmes de cellules, ainsi que les données associées que sont les matrices locales Éléments Finis.

Dans cette optique, chaque étape de la méthode multi-échelle a été implémentée dans un module indépendant des autres. Ces modules sont ensuite liés par le biais du script Python SIMULATION.PY, qui joue donc le rôle de superviseur. À chaque lancement du script, les paramètres entrés permettent de spécifier quelle étape de la méthode on souhaite effectuer. Les résultats sont ensuite stockés pour un usage ultérieure, c'est-à-dire pour l'étape suivante de la méthode multi-échelle.

Les modules développés sont :

- FCTMILIEU permet de construire les milieux modélisant les matériaux cimentaires. Il utilise le module COMBS, développé par Erwan ADAM.
- FCTVF9 permet de résoudre les problèmes de cellules par la méthode *VF9*. Une fois le système linéaire construit, il fait appel à la librairie NUMERICAL PLATON [167] pour la résolution proprement dite.
- FCTBASE construit la base d'Éléments Finis à partir des solutions des problèmes de cellules.
- CREMK calcule les matrices locales de rigidité et de masse.
- CREASSEMBLAGE assemble et résout le problème grossier par une méthode d'Éléments Finis. La résolution du système linéaire se fait par le biais de la librairie NUMERICAL PLATON.
- CALSOLUTIONMICRO reconstruit, à partir de la solution grossière, la solution fine sur tout ou partie du domaine de travail Ω.
- CALHOMOGENE regroupe plusieurs routines permettant de calculer la diffusivité équivalente du milieu, telle que définie par (1.4).

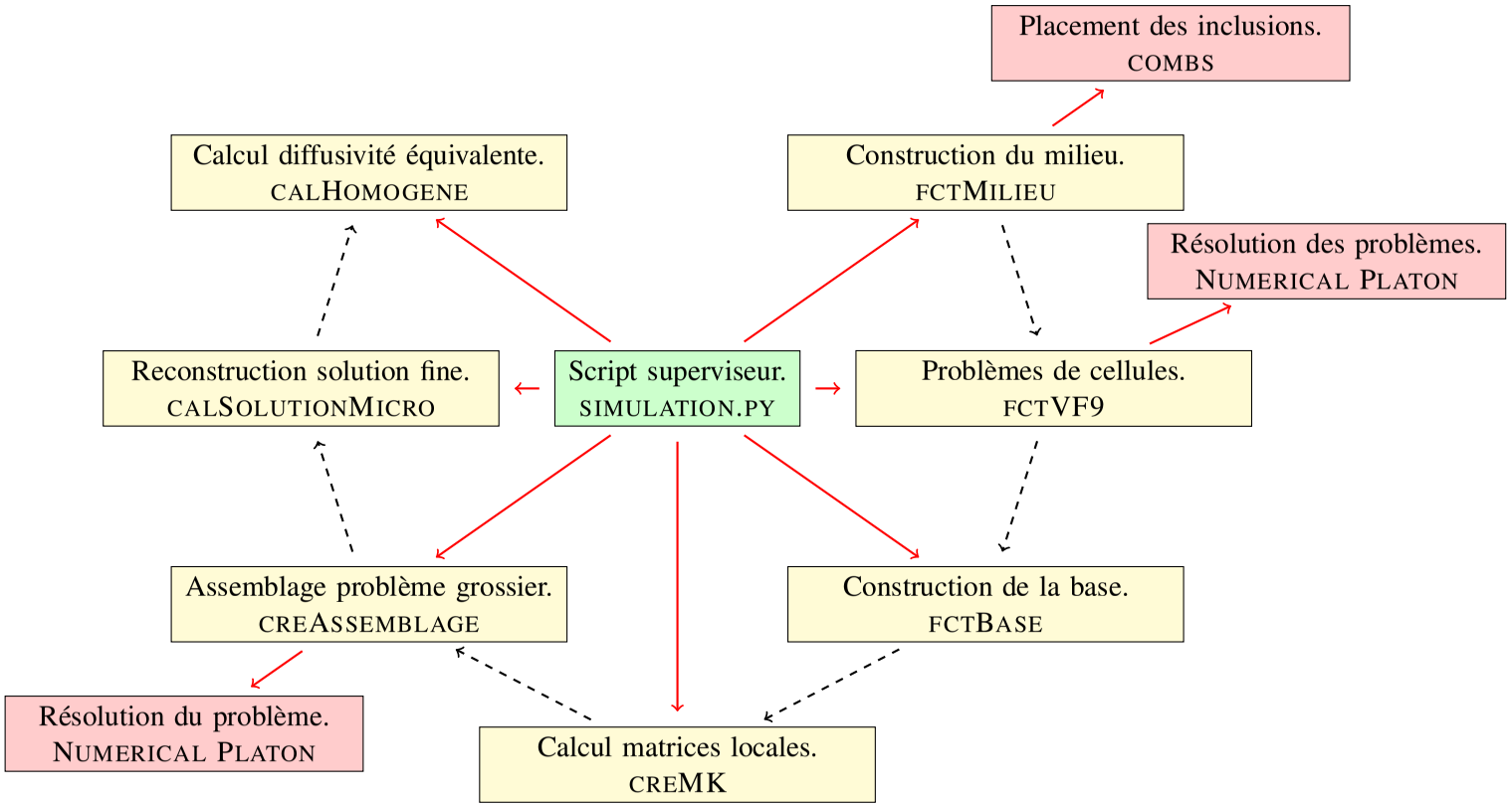


FIGURE 4.1 – Organigramme simplifié de la maquette Python implémentant la méthode Q_1/*VF9*. Les modules extérieurs au modèle sont représentés en rouge, ceux implémentés au cours de ces travaux sont en jaune. On représente en pointillés la *chaîne de calculs* qui forme la méthode multi-échelle. Les traits continus représentent une dépendance entre modules.

La Figure 4.1 présente l'organigramme liant ces modules. Y apparaissent en jaune les modules développés spécifiquement pour la méthode multi-échelle. Les modules indépendants, fruits d'un travail extérieur à ces travaux de thèse, sont représentés en rouge.

Dans la suite de cette section, on s'applique à valider l'implémentation de la méthode Q_1/*VF9*. On se concentre sur trois modules en particulier : FCTVF9, CREMK et CREASSEMBLAGE. Le module FCTMILIEU est explicité en détail à la section §4.3, en même temps que la modélisation des matériaux cimentaires. L'implémentation des autres modules ne révèle pas de difficultés particulières.

4.1.1 Qualification du module FCTVF9.

On valide dans cette section l'implémentation du schéma *VF9*. On résout pour cela plusieurs exemples du problème (3.8), que l'on reporte ici :

$$\left\{ \begin{array}{rcll} -\nabla \cdot (A \nabla u) & = & 0 & \text{dans } \Delta, \\ u & = & u_D & \text{sur } \partial\Delta. \end{array} \right. \tag{4.1}$$

Δ est ici le domaine $]0,1[\times]0,1[$. u_D représente les conditions de Dirichlet du problème.

Pour chaque exemple, on travaille sur une série de discrétisations $(T_h(\Delta))_{h>0}$, où $T_h(\Delta)$ désigne un découpage du domaine Δ en carrés de côté h. On présente alors l'erreur L^2 discrète entre u_h la solution du problème approchée sur $T_h(\Delta)$ et u la solution exacte du problème (4.1). Si on note x_k le barycentre de l'élément $k \in T_h(\Delta)$, cette erreur est définie par :

$$e_2(u_h) = h \sqrt{\sum_{k \in T_h(\Delta)} (u_h(k) - u(x_k))^2} \tag{4.2}$$

Pour exprimer les taux de convergence, on utilise la notation de Landau $\mathcal{O}(h)$ pour désigner une fonction d'ordre h quand h tend vers 0.

La résolution des systèmes linéaires s'effectue *via* la librairie *Numerical Platon* [167], elle-même appelant la bibliothèque *PetsC* [154]. Plus précisement, on utilise le solveur itératif GMRES [161, 162] assorti d'un préconditionneur ILU [65].

Exemple 4.1 *On considère l'exemple défini par :*

$$A = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$$
$$u_D = xy$$

où a et b sont des réels strictements positifs. La solution du problème est alors :

$$u = xy$$

Le schéma *VF9* étant linéaire exact, on retrouve ici la solution exacte, à la précision machine près.

Exemple 4.2 *On considère l'exemple défini par :*

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$
$$u_D = \sin(\pi x)\sinh(\pi y)$$

La solution du problème est alors :

$$u = \sin(\pi x)\sinh(\pi y)$$

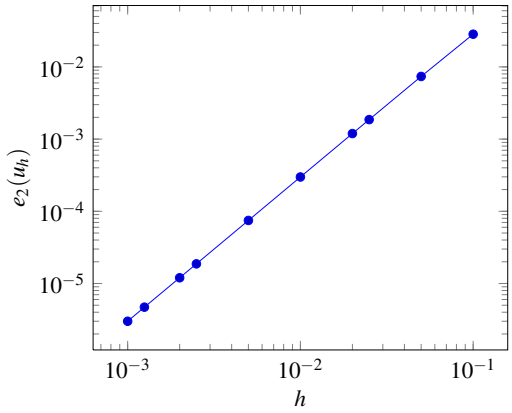


FIGURE 4.2 – Exemple 4.2 : représentation de $e_2(u_h)$ en fonction de h.

Les résultats de ce test sont présentés à la Figure 4.2. Ils montrent une convergence en $\mathcal{O}(h^2)$, soit un ordre au-dessus de la convergence théorique de la méthode Volumes Finis *VF9*. Cette *super convergence* est habituelle lors de l'utilisation des Volumes Finis *VF9* sur un milieu homogène [97]. Il ne s'agit cependant pas d'une particularité de la méthode *VF9* mais plutôt des Volumes Finis en général. Le phénomène de super convergence est en effet observable chez un grand nombre de méthodes Volumes Finis [114].

Exemple 4.3 *On considère l'exemple défini par :*

$$A = \begin{bmatrix} x^2 - y^2 + 3 & 0 \\ 0 & x^2 - y^2 + 3 \end{bmatrix}$$
$$u_D = \frac{1}{2}\ln(x^2 - y^2 + 3)$$

La solution du problème est alors connue :

$$u = \frac{1}{2}\ln(x^2 - y^2 + 3)$$

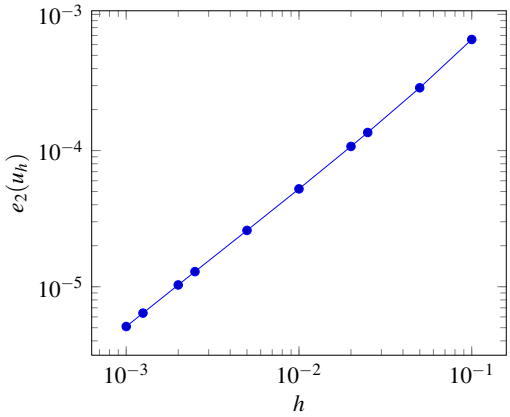


FIGURE 4.3 – Exemple 4.3 : représentation de $e_2(u_h)$ en fonction de h.

Les résultats de ce test sont présentés à la Figure 4.3. Il y apparaît clairement que la solution approchée converge vers la solution théorique, et ce avec un taux d'erreur en $\mathcal{O}(h)$.

En conclusion des bons résultats obtenus sur ces trois exemples, on peut considérer que la méthode *VF9* a été correctement implémentée au sein du module FCTVF9.

4.1.2 Qualification du module CREMK.

On valide dans cette section l'implémentation du module CREMK qui calcule les différents termes des matrices locales Éléments Finis : la matrice de raideur $\mathbb{K}_K$, la matrice de masse $\mathbb{M}_K$ et la matrice de masse de bord $\mathbb{M}_{b,F}$, pour un élément $K \in T_H(\Omega)$ et une face $F \in \Gamma_{N,H}$ donnés. Ces quantités sont définies par :

$$\forall 1 \leq i,j \leq 4 \qquad \mathbb{K}_K(i,j) = \frac{1}{2} \sum_{f \in \partial K} \left(\Phi_K^i(f) \int_f D\nabla\Phi_K^j \cdot \mathbf{n} + \Phi_K^j(f) \int_f D\nabla\Phi_K^i \cdot \mathbf{n} \right) \tag{4.3}$$

$$\forall 1 \leq i,j \leq 4 \qquad \mathbb{M}_K(i,j) = \sum_{k \in K} |k| {\Phi_K}^i(k) {\Phi_K}^j(k) \tag{4.4}$$

$$\forall 1 \leq i,j \leq 4 \quad \mathbb{M}_{b,F}(i,j) = \sum_{f \in F} |f| \Phi_K^i(f) \Phi_K^j(f) \tag{4.5}$$

où $(\Phi_K^i)_{1 \leq i \leq 4}$ est la base d'Éléments Finis construite à partir des fonctions $(\Psi_K^i)_{1 \leq i \leq 4}$, solutions sur $\hat{K}$ des problèmes de cellules. Une description précise de ces équations et un rappel des différentes notations est disponible à la section §3.3.1.

Sur une face $f \in \partial K$, le calcul de la matrice de raideur et la matrice de masse de bord fait intervenir deux quantités : $(\Phi_K^i(f))_{1\leq i\leq 4}$ les valeurs interpolées en f et $(\int_f D\nabla\Phi_K^i \cdot \mathbf{n})_{1\leq i\leq 4}$ les flux à travers f. La définition de ces deux quantités est étroitement liée à la position de f vis-à-vis de la cellule $\hat{K}$, le domaine de résolution des problèmes de cellules. Il est donc nécessaire de tester l'implémentation de ce module dans le cas où f est une face de la frontière $\partial\hat{K}$, c'est-à-dire quand le taux de sur-échantillonnage ρ est nul, et dans le cas où f est une face intérieure à la cellule $\hat{K}$, i.e. ρ est strictement positif.

Exemple 4.4 *On considère dans cet exemple que :*

$$\begin{aligned} D &= \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \\ K &=]0,1[\times]0,1[\\ F &= \{y = 0\} \end{aligned}$$

où a et b sont des réels strictements positifs. Dans ce cas, les fonctions $(\Phi_K^i)_{1\leq i\leq 4}$ *sont explicitement connues, et ce pour toute valeur de* ρ *:*

$$\begin{aligned} \Phi_K^1 &= (1-x)(1-y) \\ \Phi_K^2 &= x(1-y) \\ \Phi_K^3 &= (1-x)y \\ \Phi_K^4 &= xy \end{aligned}$$

Les matrices $\mathbb{K}_K$, $\mathbb{M}_K$ et $\mathbb{M}_{b,F}$ sont symétriques. Étant donnée la symétrie des fonctions $(\Phi_K^i)_{1\leq i\leq 4}$, on a de plus :

$$\begin{aligned} &\mathbb{K}_K(1,1) = \mathbb{K}_K(2,2) = \mathbb{K}_K(3,3) = \mathbb{K}_K(4,4), \\ &\mathbb{K}_K(1,2) = \mathbb{K}_K(1,3) = \mathbb{K}_K(2,4) = \mathbb{K}_K(3,4), \\ &\mathbb{K}_K(1,4) = \mathbb{K}_K(2,3), \end{aligned}$$

des formules semblables existant pour les matrices $\mathbb{M}_K$ et $\mathbb{M}_{b,F}$. Les matrices locales n'ont donc que trois éléments différents, correspondant aux coefficients $(1,1)$, $(1,2)$ et $(1,4)$.

Les Figures 4.4, 4.5 et 4.6 présentent l'erreur entre les valeurs théoriques et numériques des matrices locales sur ces trois coefficients. Il apparaît des simulations que les erreurs décroissent en $\mathcal{O}(h^2)$. Le calcul de la base locale $(\Phi_K^l)_{1\leq l\leq n}$ à partir des fonctions $(\Psi_{\hat{K}}^l)_{1\leq l\leq n}$ étant linéaire exact, l'erreur est indépendante de ρ.

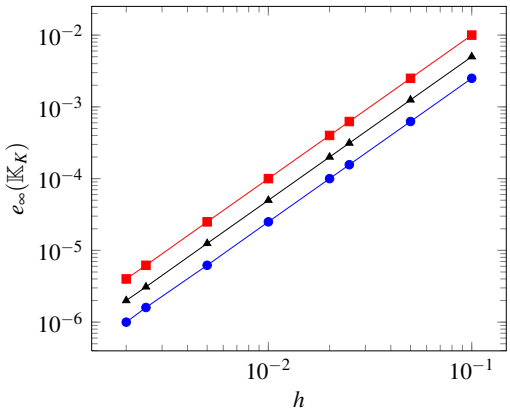


FIGURE 4.4 – Exemple 4.4 : erreur de calcul, en fonction de h, des coefficients $\mathbb{K}_K(1,1)$ (—●—), $\mathbb{K}_K(1,2)$ (—■—) et $\mathbb{K}_K(1,4)$ (—▲—). L'erreur décroît en $\mathcal{O}(h^2)$ et est indépendante de ρ.

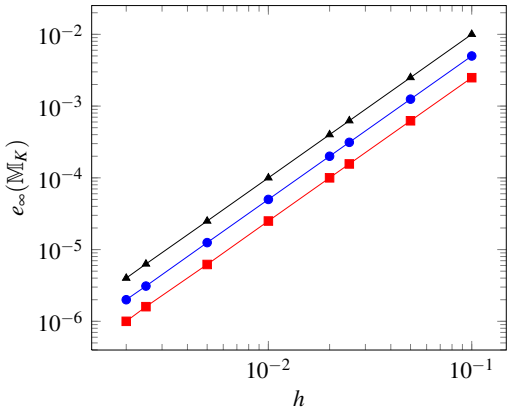


FIGURE 4.5 – Exemple 4.4 : erreur de calcul, en fonction de h, des coefficients $\mathbb{M}_K(1,1)$ (—●—), $\mathbb{M}_K(1,2)$ (—■—) et $\mathbb{M}_K(1,4)$ (—▲—). L'erreur décroît en $\mathcal{O}(h^2)$ et est indépendante de ρ.

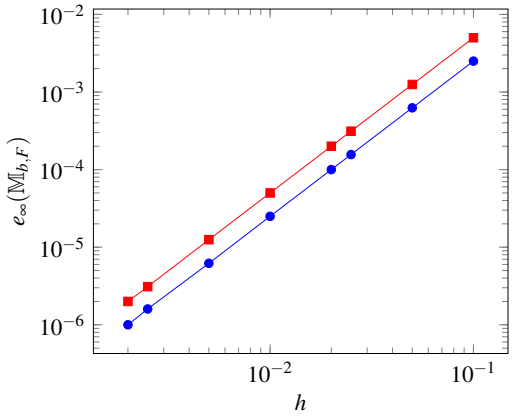


FIGURE 4.6 – Exemple 4.4 : erreur de calcul, en fonction de h, des coefficients $\mathbb{M}_{b,F}(1,1)$ (—●—) et $\mathbb{M}_{b,F}(1,2)$ (—■—). L'erreur décroît en $\mathscr{O}(h^2)$ et est indépendante de ρ.

Exemple 4.5 *On reprend dans cet exemple la solution de l'Exemple 4.2.*

$$D = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$
$$K =]0,1[\times]0,1[$$
$$F = \{y = 1\}$$
$$\beta = \sin(\pi x)\sinh(\pi y)$$

La solution du problème est alors :

$$\Psi = \sin(\pi x)\sinh(\pi y)$$

On peut alors calculer les termes (4.3)-(4.5) en posant $\Phi_K^i = \Phi_K^j = \Psi$. On présente à la Figure 4.7 l'erreur entre valeurs théoriques et numériques de ces trois termes en fonction de h. Les coefficients $\mathbb{K}(\Psi,\Psi)$ et $\mathbb{M}(\Psi,\Psi)$ convergent respectivement en $\mathscr{O}(h)$ et $\mathscr{O}(h^2)$. Le coefficient $\mathbb{M}_b(\Psi,\Psi)$ est calculé de manière exacte.

4.1.3 Qualification du module CREASSEMBLAGE.

On valide dans cette section l'implémentation du module CREASSEMBLAGE qui assemble et résout le problème d'Éléments Finis grossier. Il ne s'agit pas ici de tester la méthode multi-échelle Q_1/*VF9* dans son intégralité, mais seulement d'assurer que la méthode d'Éléments Finis a été correctement implémentée. Pour cela, on suppose que la diffusivité D est égale à la matrice *Identité*, ce qui correspond au cadre de l'Exemple 4.4 avec $(a,b) = (1,1)$. La base $(\Phi^I)_{I\in\mathscr{I}}$ est alors explicitement connue, et correspond aux Éléments Finis Q_1 classiques.

On va maintenant résoudre plusieurs exemples du problème de diffusion suivant :

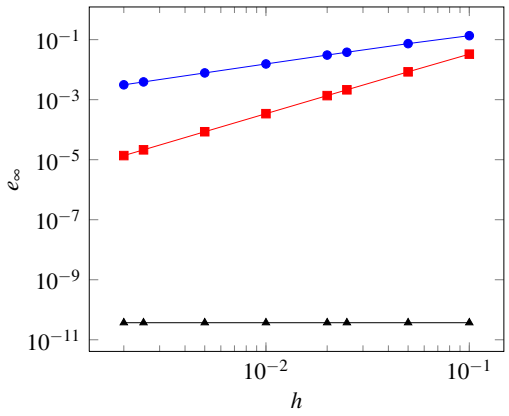


FIGURE 4.7 – Exemple 4.5 : erreur de calcul, en fonction de h, des coefficients $\mathbb{K}(\Psi,\Psi)$ (—●—), $\mathbb{M}(\Psi,\Psi)$ (—■—) et $\mathbb{M}_b(\Psi,\Psi)$ (—▲—) pour $\rho = 0$.

$$\begin{cases} -\Delta C &= f & \text{dans } \Omega, \\ C &= g_D & \text{sur } \Gamma_D, \\ D\nabla C \cdot n &= g_N & \text{sur } \Gamma_N, \end{cases} \tag{4.6}$$

où (Γ_D, Γ_N) est une partition de $\Gamma = \partial\Omega$ la frontière du domaine Ω. f, g_D et g_N sont respectivement le terme source, la condition aux limites de Dirichlet et la condition aux limites de Neumann.

Dans tout ce qui suit, on suppose que $\Omega =]0,1[\times]0,1[$. La discrétisation $T_H(\Omega)$ de Ω se compose de macroéléments carrés de longueur H.

Exemple 4.6 *On considère dans cet exemple que :*

$$\begin{aligned} f &= 0 \\ \Gamma_D &= \partial\Omega \\ g_D &= \sin(\pi x)\sinh(\pi y) \end{aligned}$$

La solution du problème est alors :

$$C = \sin(\pi x)\sinh(\pi y)$$

Exemple 4.7 *On considère dans cet exemple que :*

$$\begin{aligned} f &= 0 \\ \Gamma_D &= \{0,1\}\times[0,1] \\ g_D &= \sin(\pi x)\sinh(\pi y) \\ \Gamma_N &= [0,1]\times\{0,1\} \\ g_N &= \frac{\partial}{\partial \mathbf{n}}(\sin(\pi x)\sinh(\pi y)) \end{aligned}$$

La solution du problème est alors :

$$C = \sin(\pi x)\sinh(\pi y)$$

On choisit volontairement de se restreindre en n'imposant les conditions de Neumann qu'en deux bords disjoints de Ω afin de s'affranchir des problèmes informatiques liés aux coins du domaine.

Exemple 4.8 *On considère dans cet exemple que :*

$$\begin{aligned} f &= \sin(\pi x)\sin(\pi y) \\ \Gamma_D &= \partial\Omega \\ g_D &= 0 \end{aligned}$$

La solution du problème est alors :

$$C = \frac{1}{2\pi^2}\sin(\pi x)\sin(\pi y)$$

Le critère de validation choisit ici est l'erreur maximale entre valeurs calculées et théoriques aux noeuds $I \in \mathscr{I}$:

$$e_\infty(C_H) = \max_{I\in\mathscr{I}} |C_H(I) - C(I)| \tag{4.7}$$

Il ne s'agit pas du maximum de l'erreur entre solution théorique et calculée car on ne recalcule pas la solution $C_{H,h}$ au niveau fin. De cette manière, la discrétisation des macroéléments K n'influence le résultat que par le biais des matrices de raideurs et de masses, influence que l'on néglige dans cette partie.

On présente à la Figure 4.8 les courbes d'erreurs des Exemples 4.6, 4.7 et 4.8 en fonction de H.

4.2 Validation de la méthode Q_1/*VF9* sur un exemple théorique.

4.2.1 Convergence théorique.

On se place dans cette section dans le cadre de l'homogénéisation théorique périodique. Soit d une fonction périodique sur $Y = [0,1]^2$, et $\varepsilon > 0$ un paramètre d'échelle, D est défini par :

$$D^\varepsilon = d(\frac{x}{\varepsilon})$$

Le problème de diffusion à résoudre est alors :

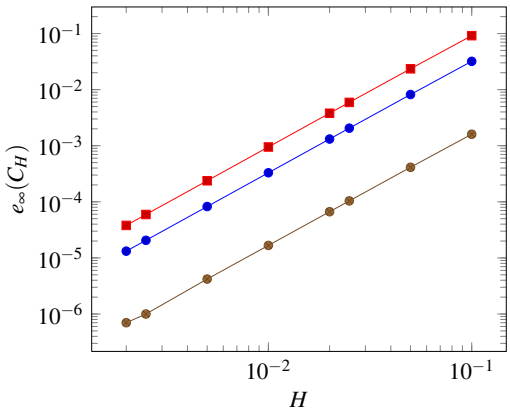


FIGURE 4.8 – Erreur $e_\infty(C_H)$ en fonction de H pour les Exemples 4.6 (—●—), 4.7 (—■—) et 4.8 (—●—).

$$\begin{cases} -\nabla\cdot(D^\varepsilon\,\nabla C^\varepsilon) &= f & \text{dans } \Omega, \\ C^\varepsilon &= g_D & \text{sur } \Gamma_D, \\ D^\varepsilon\,\nabla C^\varepsilon\cdot\mathbf{n} &= g_N & \text{sur } \Gamma_N, \end{cases} \tag{4.8}$$

Comme on l'a définit au chapitre §3, (Γ_D,Γ_N) forme une partition de $\Gamma=\partial\Omega$ la frontière du domaine Ω. f, g_D et g_N sont respectivement le terme source, la condition aux limites de Dirichlet et la condition aux limites de Neumann du problème.

Dans ce cadre, il est possible d'estimer l'erreur entre C^ε, la solution théorique du problème (4.8) et $C^\varepsilon_{H,h}$, la solution numérique obtenue par la méthode multi-échelle Q_1/*VF9* [28, 90, 120, 121].

Pour $h<\varepsilon<H$, on a :

$$\|C^\varepsilon - C^\varepsilon_{H,h}\|_2 \le A_1\left(H+\sqrt{\frac{\varepsilon}{H}}+\frac{h}{\varepsilon}\right) \qquad \text{si } \rho=0, \tag{4.9}$$

$$\|C^\varepsilon - C^\varepsilon_{H,h}\|_2 \le A_2\left(H+\frac{\varepsilon}{H}+\frac{h}{\varepsilon}\right) \qquad \text{si } \rho>0. \tag{4.10}$$

A_1 et A_2 sont des constantes ne dépendant d'aucun des paramètres H, ε et h.

L'influence du taux de sur-échantillonnage ρ sur A_2 est mal connu. Les travaux de *Efendiev* et al. [90] donnent à penser qu'on peut réécrire (4.10) sous la forme :

$$\|C^\varepsilon - C^\varepsilon_{H,h}\|_2 \le A_{2,1}H+\frac{A_{2,2}}{\rho}\frac{\varepsilon}{H}+A_{2,3}\frac{h}{\varepsilon} \tag{4.11}$$

mais le problème reste le même, puisque la dépendance en ρ des constantes $A_{2,1}$, $A_{2,1}$ et $A_{2,1}$ n'est pas connue. Au vu des simulations numériques sur le sujet, il semble cependant que $A_{2,1}$, $A_{2,1}/\rho$ et $A_{2,1}$ tendent vers des constantes non nulles quand ρ augmente [90, 120, 121].

Pour étudier la convergence de la méthode Q_1/*VF9*, il est donc nécessaire de faire tendre H, ε et h vers 0, tout en supposant que l'on a :

$$h \ll \varepsilon \ll H \tag{4.12}$$

Il est très difficile de remplir numériquement l'ensemble de ces conditions. C'est pourquoi les résultats présentés dans les sections suivantes se concentrent sur l'évolution de l'erreur quand *un seul* paramètre tend vers 0, les autres étant constants. Les courbes présentées ne sont donc en aucun cas des courbes de *convergence*.

4.2.2 Validation.

On présente ici les résultats de la méthode Q_1/*VF9* sur un exemple théorique simple. Le but premier est ici de retrouver le comportement attendu de la méthode des Éléments Finis multi-échelles, tel que décrit à la section précédente. C'est pourquoi le milieu étudié ne présente qu'une variation modérée de la diffusivité : il n'est pas question ici de tester la robustesse de la méthode.

Exemple 4.9 *On résout sur* $\Omega = [0,1]^2$ *le problème suivant :*

$$\left\{ \begin{array}{rcll} -\nabla(D\nabla C) & = & f & \textit{dans } \Omega, \\ C & = & 0 & \textit{sur } \Gamma_D = \partial\Omega. \end{array} \right. \tag{4.13}$$

f *est une fonction* en créneau*, c'est-à-dire égale à* 1 *au sein de la sphère de centre* $(x_0, y_0) = (0.2, 0.4)$ *et de rayon* 0.05 *et nulle partout ailleurs.*

Avec $p = 50$*, on pose :*

$$D(x,y) = (1.1 + \sin(2\pi px)\sin(2\pi py))\, I_d \tag{4.14}$$

Le champs de diffusivité est représenté sur une portion du domaine Ω *à la Figure 4.9 (à gauche).*

On divise le milieu Ω en $M = m \times m$ macro-éléments K carrés de côté $H = 1/m$, formant la discrétisation grossière $T_H(\Omega)$. Chaque macro-élément K est ensuite divisé en $N = n \times n$ micro-éléments carrés de côté $h = 1/(mn)$, formant la discrétisation $T_h(K)$. Avec ρ le taux de sur-échantillonnage, la discrétisation $T_h(\hat{K})$ d'une cellule $\hat{K}$ comporte alors $(1+2\rho)^2 n^2$ micro-éléments.

Afin d'obtenir une solution de référence C_r, on résout le problème (4.13) *via* une méthode directe de Volumes Finis à 4 millions d'éléments. En posant $h_r = 5.10^{-4}$, la discrétisation $T_{h_r}(\Omega)$ du milieu Ω comporte donc 2000×2000 micro-éléments. Une représentation de C_r est présentée à la Figure 4.9 (à droite).

On compare ensuite les solutions numériques $C_{H,h}$ de l'Exemple 4.9, comportant $m^2 \times n^2$ micro-éléments, à C_r. Soit $k_r \in T_{h_r}(\Omega)$, pour tout $h > h_r$, on définit $\Upsilon_h(k_r)$ l'ensemble des éléments $k \in T_h(K)$ ayant une intersection non-vide avec k_r :

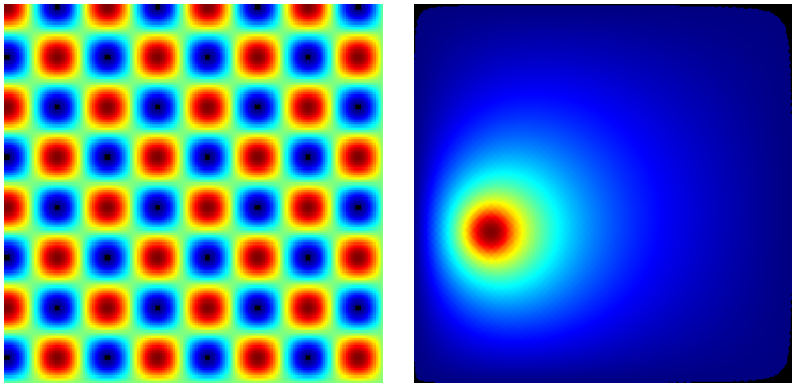


FIGURE 4.9 – Champs de diffusivité (zoom) et solution de référence C_r pour l'Exemple 4.9.

$$\Upsilon_h(k_r) = \{k \in T_h(K), K \in T_H(\Omega), k \cap k_r \neq \emptyset\} \tag{4.15}$$

On note $n_h(k_r)$ le cardinal de $\Upsilon_h(k_r)$. L'erreur L_2 discrète est alors calculée par :

$$e_2(C_{H,h}) = h_r \sqrt{\sum_{k_r \in T_{h_r}(\Omega)} \left[C_r(k_r) - \frac{1}{n_h(k_r)} \sum_{k \in \Upsilon_h(k_r)} C_{H,h}(k) \right]^2} \tag{4.16}$$

On peut définir une quantité équivalente sur la densité de flux $D\nabla C_{H,h}$:

$$e_2(D\nabla C_{H,h}) = h_r \sqrt{\sum_{k_r \in T_{h_r}(\Omega)} \left[D_r \nabla C_r(k_r) - \frac{1}{n_h(k_r)} \sum_{k \in \Upsilon_h(k_r)} D_h \nabla C_{H,h}(k) \right]^2} \tag{4.17}$$

La Figure 4.10 présente les courbes d'erreurs sur la concentration $C_{H,h}$ et la densité de flux $D\nabla C_{H,h}$ en fonction de h, pour différentes valeurs fixées de H. On note que les deux erreurs stagnent rapidement. Diminuer sans fin le pas de résolution des problèmes locaux est donc inutile si le nombre de degré de liberté du maillage grossier (i.e $m = 1/H$) est limité. Dans les deux cas présentés, on note que $h = 1e-3$ suffit amplement à atteindre la zone limite.

Bien que les erreurs en elles-mêmes augmentent avec ρ, le comportement générale de l'erreur reste lui pratiquement inchangé. .

Sur la Figure 4.11, on montre cette fois l'évolution de l'erreur quand H diminue, en imposant $h = 0.02H$ (à gauche) ou $h = 0.01H$ (à droite). Il apparaît que l'erreur n'est pas une fonction décroissante de H, mais qu'elle passe par un minimum avant de remonter. En effet, si H devient trop petit, la taille des problèmes de cellules

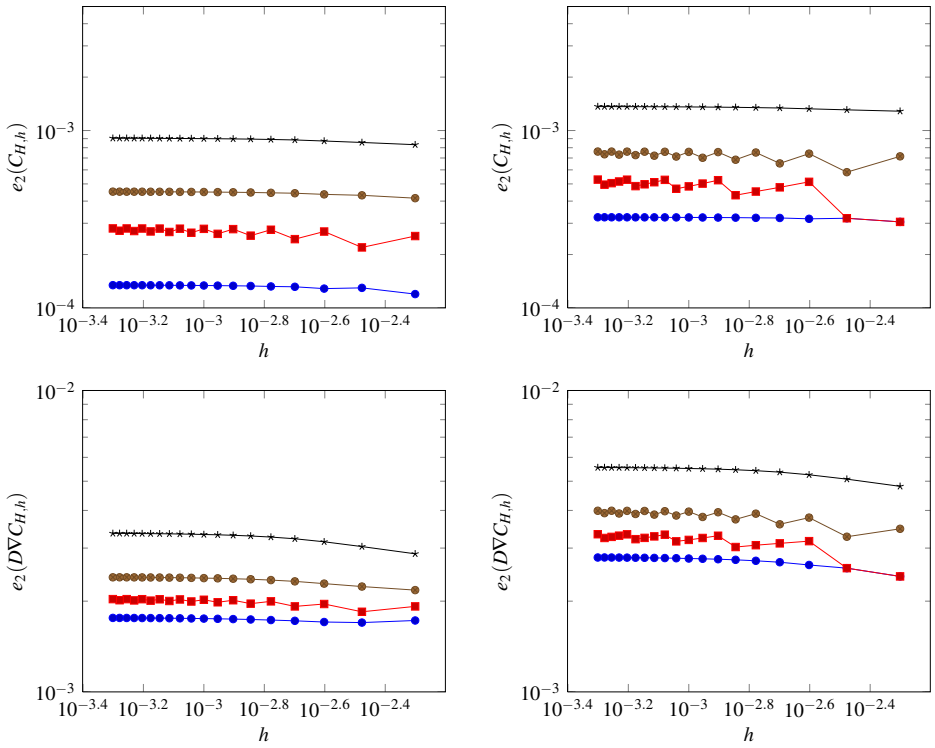


FIGURE 4.10 – Exemple 4.9 : évolution des erreurs $e_2(C_{H,h})$ (en haut) et $e_2(D\nabla C_{H,h})$ (en bas) pour $H = 0.01$ (gauche) et $H = 0.05$ (droite). Sur chaque courbe, on a tracé l'erreur en fonction de h pour quatre valeurs de ρ : 0 (—●—), 0.05 (—■—), 0.10 (—●—) et 0.20 (—⋆—).

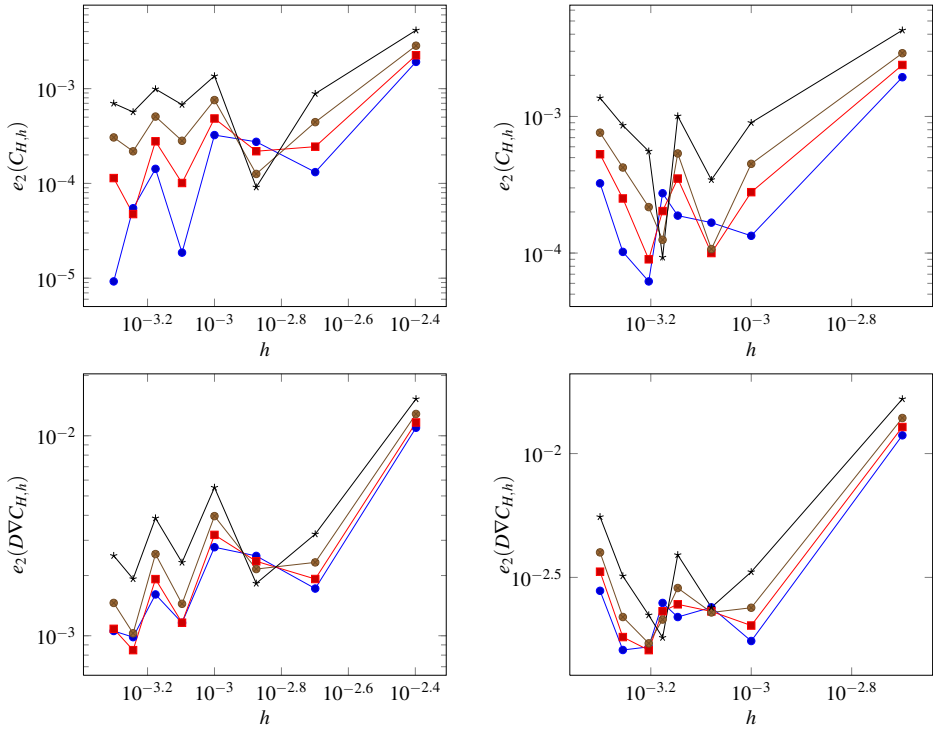


FIGURE 4.11 – Exemple 4.9 : évolution des erreurs $e_2(C_{H,h})$ (en haut) et $e_2(D\nabla C_{H,h})$ (en bas) pour $h = 0.02H$ (gauche) et $h = 0.01H$ (droite). Sur chaque courbe, on a tracé l'erreur en fonction de h pour quatre valeurs de ρ : 0 (—●—), 0.05 (—■—), 0.10 (—●—) et 0.20 (—⋆—).

devient trop petite par rapport aux oscillations du milieu et l'homogénéisation de la cellule n'est plus possible. Il faut en effet que chaque cellule englobe 4 à 5 cellules périodiques du milieu pour que la méthode soit efficace. Dans le cas de l'Exemple 4.9, cela revient à choisir H supérieur à 0.08.

Remarque 4.1 *Comme on l'a expliqué à la section §4.2.1, les courbes présentées ici ne sont pas des courbes de* convergence *de la méthode* Q_1/VF9*, mais des courbes d'*évolution *de l'erreur en fonction d'un paramètre (h, H ou* $\alpha = h/H$*), les autres étant fixés.*

Ces simulations valident correctement l'implémentation Python de la méthode Q_1/*VF9*. Si on fait varier l'une ou l'autre des deux valeurs h et H, l'erreur diminue jusqu'à atteindre un minimum qui dépend du second paramètre resté fixe. Le sur-échantillonnage ρ, quant à lui, permet de diminuer les valeurs maximales des erreurs, mais au prix d'une moins bonne précision, surtout lorsque le milieu est grossièrement discrétisé («grandes» valeurs de H et h.) Lorsque H et h sont suffisament petits,par contre, les résultats sont légèrement en faveur du sur-échantillonnage.

4.3 Application aux matériaux cimentaire.

4.3.1 Génération d'un matériau cimentaire *via* le module COMBS.

Comme on l'a décrit à la section §3.6, les matériaux cimentaires sont modélisés dans le cadre des travaux de cette thèse par un ensemble d'inclusions de formes diverses enchâssées dans un milieu homogène. Pour construire un échantillon d'un matériau cimentaire, il est donc nécessaire de placer ces inclusions dans le milieu. Un algorithme spécifique, le module COMBS a été développé par *Erwan Adam* afin de s'acquitter de cette tâche.

Soit une liste détaillée, précisant la taille, la forme et le nombre des inclusions voulues, le module COMBS va placer aléatoirement ces inclusions au sein du domaine. En sortie du module, on obtient un fichier décrivant chaque inclusion et sa position. À chaque inclusion est également associé le nom de la phase à laquelle elle appartient.

Il est possible de préciser plusieurs options, en fonction du milieu que l'on souhaite générer. On peut ainsi demander au module de construire un milieu périodique ; les inclusions débordant d'un coté du domaine réapparaîssant de l'autre ; ou au contraire interdire aux inclusions d'approcher les frontières du domaine. Il est de plus possible de paramétrer la distance minimale séparant deux inclusions.

Enfin, il arrive que le module n'arrive pas à placer toutes les inclusions voulues, soit parce que le domaine est plein, soit parce que la place libre est «difficile» à trouver. Dans ce dernier cas, l'expérience montre que l'on peut espérer atteindre un taux de remplissage de près de 80% en volume si on laisse suffisament de temps à l'algorithme (généralement plus de 12h).

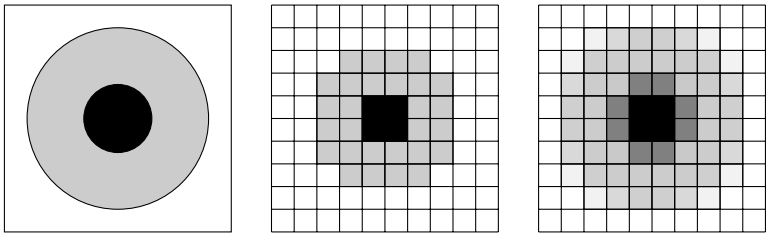


FIGURE 4.12 – Pour un milieu donné (à gauche), deux méthodes d'attribuer une diffusivité aux éléments du maillage fin : par *dichotomie* (au milieu) et par *moyenne* (à droite).

La première version de ce module ne permettait que de placer des inclusions sphériques à une ou plusieurs couches [21]. C'est cette version qui est utilisée, dans le cadre $2D$, pour générer les milieux utilisés par l'implémentation présentée dans ce chapitre. Les dernières versions, en faisant appel à la plate-forme *SALOME* [163], permettent maintenant de placer des formes quelconques en deux et trois dimensions [70].

4.3.2 Discrétisation du milieu.

4.3.2.1 Principes.

Comme on l'a précisé au début de ce chapitre, la maquette Python présentée ici travaille exclusivement avec des maillages rectangulaires réguliers en dimension deux aux deux niveaux : grossier et fin. A l'échelle fine, une cellule $\hat{K}$ rectangulaire est donc découpée en rectangles k et une unique valeur de diffusivité est attribuée à chaque rectangle. Cette discrétisation du milieu est très pratique, puisque les connectivités entre les divers éléments (sommets/éléments, éléments/éléments, arêtes/éléments ...) sont explicitement connus. Un problème se pose cependant dans le cas où l'on veut discrétiser des inclusions sphériques ou en couronne par un ensemble de rectangles. La question est de savoir quelle valeur de diffusivité attribuer aux rectangles à cheval sur la *matrice* et une, voire plusieurs, inclusions.

Dans un premier temps, on a choisi de départager le maillage par *dichotomie* : seuls les rectangles inclus dans une inclusion se voyant attribuer la diffusivité de l'inclusion. Les autres, partiellement dans l'inclusion ou non, sont considérés comme appartenant à la matrice, et recoivent la diffusivité correspondante. Un exemple de milieu discrétisé par dichotomie est visible à la Figure 4.12, au milieu.

Cette caractérisation a pour elle le mérite de la simplicité, mais ne permet qu'une description grossière du milieu. En effet, cette méthode découpe les inclusions, les réduisant à leurs seuls rectangles internes, ce qui entraine une perte de masse. C'est particulièrement flagrant dans le cas des petites inclusions, dont la taille est de l'ordre du pas du maillage. Les inclusions se retrouvent alors mo-

délisées par un simple rectangle, ou encore, pour la taille juste supérieure, par une croix.

Il est également impossible de représenter correctement deux inclusions très proches l'une de l'autre. Quelle que soit la distance réelle entre deux inclusions, cette méthode impose une distance effective égale au pas h du maillage fin $T_h(\hat{K})$.

Pour toutes ces raisons, une discrétisation par *moyenne* a été développée. On attribue, pour chaque microélement k, une valeur de diffusivité au prorata des aires des différentes phases (matrice, inclusions) le traversant.

Soit $(\phi_i)_{1\leq i\leq p}$ l'ensemble des phases d'un rectangle k, $(D_i)_{1\leq i\leq p}$ les diffusivités correspondantes, et $(a_i)_{1\leq i\leq p}$ l'aire occupée par ces phases au sein du rectangle. On définit alors D_k la diffusivité du rectangle par moyenne arithmétique :

$$D_k = \frac{1}{\sum_{1\leq i\leq p} a_i} \sum_{1\leq i\leq p} a_i D_i \tag{4.18}$$

Un exemple de milieu discrétisé par filtrage est visible à la Figure 4.12, à droite.

Il est à noter que cette méthode n'est utilisable que dans le cadre particulièrement avantageux où l'on se trouve. Il suffit ici de déterminer les aires d'intersections entre des sphères et des rectangles, ce qui est explicité dans la suite de cette section. Le problème est autrement plus difficile en 3 dimension, ou si la forme des inclusions n'est pas aussi simple. C'est pour cette raison que cette technique n'est pas utilisée au sein de l'implémentation *MPCube*, comme on l'explique au chapitre §6.

4.3.2.2 Caractérisation de l'intersection d'une sphère et d'un rectangle.

Soit C un cercle de rayon r, R un rectangle de dimensions (h_x, h_y) et $I(R,C)$ leur intersection éventuellement vide, dont on cherche à calculer l'aire. On décompose $\mathscr{S}(R)$, l'ensemble des sommets du rectangle R, en $\mathscr{S}_i(R)$ et $\mathscr{S}_e(R)$, sous-ensembles des sommets intérieurs stricts, respectivement extérieurs large, au cercle C. On note $\mathscr{A}(R)$ l'ensemble des arêtes de R, et $\mathscr{I}(a)$ l'ensemble, éventuellement vide, des points d'intersection de l'arête $a \in \mathscr{A}(R)$ et du cercle C.

On pose :

$$q = Card(\mathscr{S}_i(R)) \tag{4.19}$$

On caractérise alors l'intersection $I(R,C)$ en fonction des valeurs de q. Dans le cas où $q = 0$, c'est-à-dire quand le cercle C ne contient aucun sommet de R, on distingue deux cas, selon que le centre de C soit dans R ou non. Au total, on distingue donc six types d'intersection qui sont présentés à la Figure 4.13.

Chaque intersection est ensuite découpée en un ensemble de triangles rectangles, de trapèzes et de cordes. Les aires de ces figures géométriques sont facilement

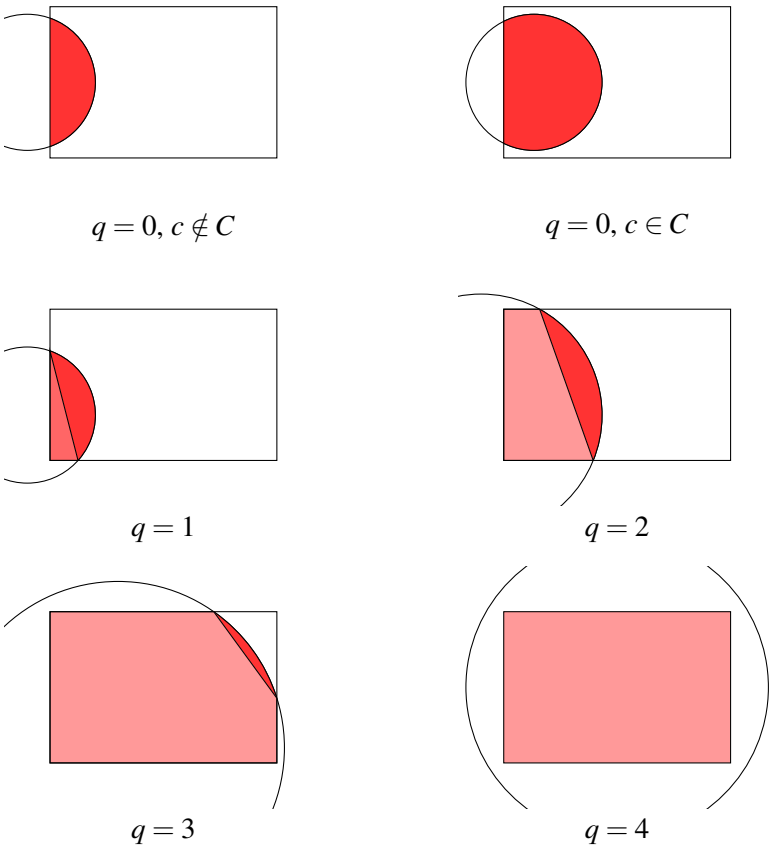


FIGURE 4.13 – Intersections d'un cercle C et d'un rectangle R en fonction du nombre q de sommets R à l'intérieur de C. Les zones délimitées par l'Algorithme 4.1 afin de calculer l'aire d'intersection apparaissent en différentes teintes de rouge.

calculables et sont rappelées à la Figure 4.14. Finalement, l'aire de l'intersection $I(R,C)$ s'obtient par sommation, ou retranchement, des aires des zones concernées.

Algorithme 4.1 Aire d'intersection entre le cercle C de rayon r et le rectangle R de côté (h_x, h_y).

Si $q = 0$ **alors**
  Si $C \in R$ **alors**
    $Aire \leftarrow \pi r^2 - \sum_{a \in \mathscr{A}(R)} C_r(r, l_i(a))$.
  sinon
    $Aire \leftarrow \sum_a C_r(r, l_i(a))$.
  Fin Si
sinon Si $q = 1$ **alors**
  $Aire \leftarrow T(l_i(a_1)), l_i(a_4)) + C_r(r,L) - C_r(r, l_i(a_2)) - C_r(r, l_i(a_4))$.
sinon Si $q = 2$ **alors**
  $Aire \leftarrow T_r(l_i(a_1), l_i(a_3), l_i(a_4)) + C_r(r,L) - C_r(r, l_i(a_2))$.
sinon Si $q = 3$ **alors**
  $Aire \leftarrow h_x h_y - T(l_e(a_1), l_e(a_2)) + C_r(r,L)$.
sinon Si $q = 4$ **alors**
  $Aire \leftarrow h_x h_y$.
Fin Si

D'un point de vue algorithmique, afin de calculer proprement ces surfaces, on commence par ordonner les sommets et les arêtes de R selon le sens trigonométrique. On fait ensuite ensuite de même avec les points $(\mathscr{I}(a))_{a \in \mathscr{A}(R)}$, en prenant pour comme origine l'arête sortante de C, c'est-à-dire l'arête $a = [S_1 S_2] \in \mathscr{A}(R)$ telle que :

$$S_1 \in \mathscr{S}_i(R) \qquad \text{et} \qquad S_2 \in \mathscr{S}_e(R) \tag{4.20}$$

Dans les cas $n = 0$ et $n = 4$, aucune arête ne satisfait cette condition. L'origine dans ces deux cas est sans importance et n'importe quelle point de départ convient donc. On définit et calcule ensuite les quantités suivantes :

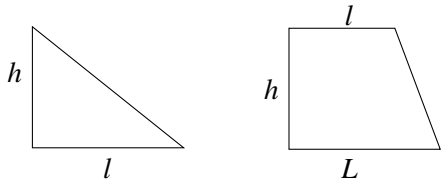
- L : distance entre le premier et le dernier point d'intersection.
- $l_i(a)$, respectivement $l_e(a)$, est la longueur de la portion de a à l'intérieur, respectivement à l'exterieur, de C.

On calcule alors l'aire de $I(R,C)$ en suivant l'Algorithme 4.1.

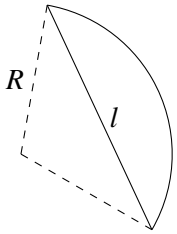
4.3.3 Un exemple de pâte cimentaire.

4.3.3.1 Définition de l'exemple.

On considère ici un milieu reprenant la physionomie de la pâte cimentaire.On a essayé de respecter au mieux les caractéristiques réelles (proportions, valeurs de



$$T(l,h) = \tfrac{1}{2}lh \qquad T_r(l,L,h) = \tfrac{1}{2}(l+L)h$$



$$C_r(R,l) = \tfrac{R}{2}\left(R\arccos(1-\tfrac{l^2}{2R^2}) - l\sqrt{1-\tfrac{l^2}{4R^2}}\right)$$

FIGURE 4.14 – Aires de surfaces élémentaires. On note $T(l,h)$, $T_r(l,h)$ et $C_r(R,l)$ les aires respectives du triangle, du trapèze et de la corde représentés ici.

diffusivité, tailles et formes des inclusions) de la pâte cimentaire, mais certaines valeurs ont été adaptées au cas bidimensionnel. C'est le cas notamment du taux de remplissage qui est inférieur à la normale. Quoi qu'il en soit, l'intérêt de cette expérience reste entier : tester la robustesse de la méthode sur un cas réaliste.

On travaille sur l'espace $\Omega = [0,1] \times [0,1]$, qui représente un carré d'$1mm^2$ de pâte cimentaire. Cet espace se compose d'un milieu homogène représentant la porosité capillaire, de diffusivité $D_0 = 2.2.10^{-9} m^2.s^{-1}$, dans lequel on place plusieurs milliers d'inclusions réparties de la façon suivante :

- 1900 Sphères homogènes ($d_3 = 9.10^{-12} m^2.s^{-1}$, $r_3 \in [0.001, 0.013]$) en LD C-S-H (*Low Density* C-S-H).
- 1700 Sphères homogènes ($d_2 = 1.10^{-12} m^2.s^{-1}$, $r_2 \in [0.001, 0.013]$) en HD C-S-H (*High Density* C-S-H).
- 3500 Sphères homogènes ($d_1 = 1.10^{-20} m^2.s^{-1}$, $r_1 \in [0.001, 0.013]$) représentant la portlandite, les aluminates et les anhydres.
- 200 Sphères multi-couches (rayon $r_4 \in [0.013, 0,028]$) composées d'un coeur de portlandite (ou d'aluminates, d'anhydre) occupant 40% du rayon, et de deux coquilles une de HD C-S-H (de $0.4r_4$ à $0.9r_4$) et une de LD C-S-H (de $0.9r_4$ à r_4).

Ces inclusions sont placées aléatoirement dans le domaine Ω par le module COMBS, décrit à la section §4.3.1.

Exemple 4.10 *On résout le problème grossier suivant :*

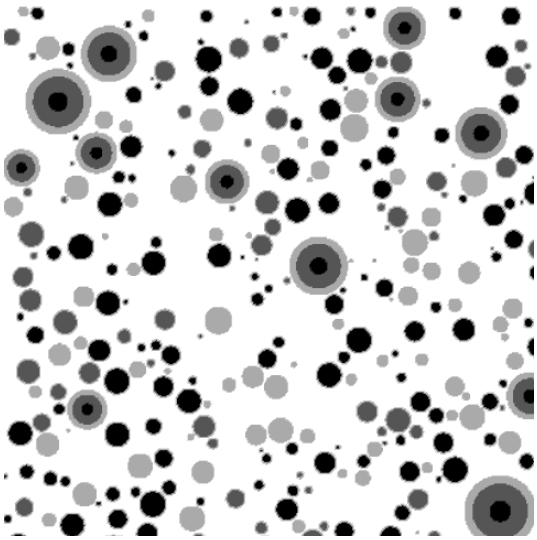


FIGURE 4.15 – Représentation de la pâte cimentaire modélisée : porosité capillaire (blanc), LD C-S-H (gris clair), HD C-S-H (gris foncé), portlandite, aluminates et anhydres (noir).

$$\left\{\begin{array}{rcll} -\nabla(D\nabla C) & = & 0 & \textit{dans } \Omega, \\ C & = & \delta_{\{x=0\}} & \textit{sur } \Gamma_D = \{0,1\}\times[0,1], \\ D\nabla C\cdot\mathbf{n} & = & 0 & \textit{sur } \Gamma_N = [0,1]\times\{0,1\}. \end{array}\right. \tag{4.21}$$

où δ représente le symbole de Kronecker.

Dans le milieu, on impose donc à la concentration d'être égale à 1 en $x = 0$ et à 0 en $x = 1$. Le flux à travers les parois verticales est nul et il n'y a pas de terme source. Si on note $L = 1mm$ la longueur entre les deux parois horizontales de Ω et A^- son arête inférieure, on définit la *diffusivité équivalente du milieu* D^* par :

$$D^* = \frac{1}{L}\int_{A^-} D\nabla C\cdot\mathbf{n}. \tag{4.22}$$

Le problème grossier et le calcul de la diffusivité équivalente imitent les mesures expérimentales de diffusivité équivalente [48, 49, 63, 73].

4.3.3.2 Simulations.

On résout numériquement l'Exemple 4.10 pour plusieurs jeux de maillages grossiers et fins. On reprend les notations de la section §4.2 : on divise le milieu Ω en $M = m\times m$ macro-éléments K carrés de côté $H = 1/m$, formant ainsi la discrétisation grossière $T_H(\Omega)$. Chaque macro-élément K est ensuite divisé en $N = n\times n$ micro-éléments carrés de côté $h = 1/(mn)$, formant la discrétisation fine $T_h(K)$.

Avec ρ le taux de sur-échantillonnage, la discrétisation fine $T_h(\hat{K})$ d'une cellule $\hat{K}$ comporte alors $(1+2\rho)^2 n^2$ micro-éléments.

Afin d'obtenir une solution de référence C_r, on résout l'Exemple 4.10 *via* une méthode directe de Volumes Finis à 4 millions d'éléments. En posant $h_r = 5.10^{-4}$, la discrétisation $T_{h_r}(\Omega)$ du milieu Ω comporte donc 2000×2000 micro-éléments.

On compare ensuite les solutions numériques $C_{H,h}$ de l'Exemple 4.10, comportant $m^2 \times n^2$ micro-éléments, à C_r. On utilise pour cela les définitions des erreurs L_2 discrètes (4.16) et (4.17), portant respectivement sur les quantités $C_{H,h}$ et $D_h \nabla C_{H,h}$.

Remarque 4.2 *Comme pour l'Exemple 4.9, les courbes présentées ici ne sont pas des courbes de* convergence *de la méthode Q_1/VF9, mais des courbes d'*évolution *de l'erreur en fonction d'un paramètre (h, H ou $\alpha = h/H$) donné, les autres étant fixés. On se reportera à la section §4.2.1 pour plus de détails à ce sujet.*

4.3.3.3 Influence de la discrétisation fine $T_h(\hat{K})$.

Les courbes présentées à la Figure 4.16 montrent l'évolution des erreurs L^2 discrètes sur la concentration ($e_2(C_{H,h})$, en haut), le gradient ($e_2(\nabla C_{H,h})$, au milieu) et le flux ($e_2(D_h \nabla C_{H,h})$, en bas) quand h diminue, pour différentes valeurs fixées de H. Deux valeurs de m sont ici représentées : $H = 0.1$ (à gauche) et $H = 0.02$ (à droite).

Pour les deux valeurs faibles du recouvrement, $\rho = 0$ (en bleu) et $\rho = 0.05$ (en rouge), les erreurs sont sensiblement égales. Tout au plus le recouvrement non nul donne-t'il des résultats légèrement meilleurs sur le gradient et le flux. Ces erreurs diminuent jusqu'à atteindre un seuil minimum qui dépend de H.

Par contre, un recouvrement plus élevé comme $\rho = 0.10$ (en marron) et surtout $\rho = 0.20$ (en noir), a une influence très négative sur la méthode. Tout d'abord, quand H est grand, les résultats sont moins bons que ceux de la méthode classique. Mais surtout l'erreur diverge quand H tend vers 0 : la méthode est instable.

4.3.3.4 Influence de la discrétisation grossière $T_H(\Omega)$.

On représente à la Figure 4.16 l'évolution des erreurs L^2 discrètes sur la concentration ($e_2(C_{H,h})$, en haut), le gradient ($e_2(\nabla C_{H,h})$, au milieu) et le flux ($e_2(D_h \nabla C_{H,h})$, en bas) quand h diminue, pour un rapport $\alpha = h/H$ fixés. Deux rapports sont ici représentées : $\alpha = 0.02$ (à gauche) et $\alpha = 0.01$ (à droite).

On note que, pour la concentration et le gradient, le cas sans recouvrement (en bleu) stagne très rapidement à un niveau minimum dépendant de α. Le cas $\rho = 0.05$ (en rouge) suit la même allure, mais avec des valeurs légèrement inférieures. Le cas $\rho = 0.10$ (en marron) donne un résultat mitigé, précurseur des très mauvais résultats du cas à fort recouvrement $\rho = 0.20$ (en noir). Dans ce dernier cas en effet, la méthode diverge.

Si on se penche maintenant sur le flux, on note que la méthode stagne, et ce quel que soit le taux de sur-échantillonnage.

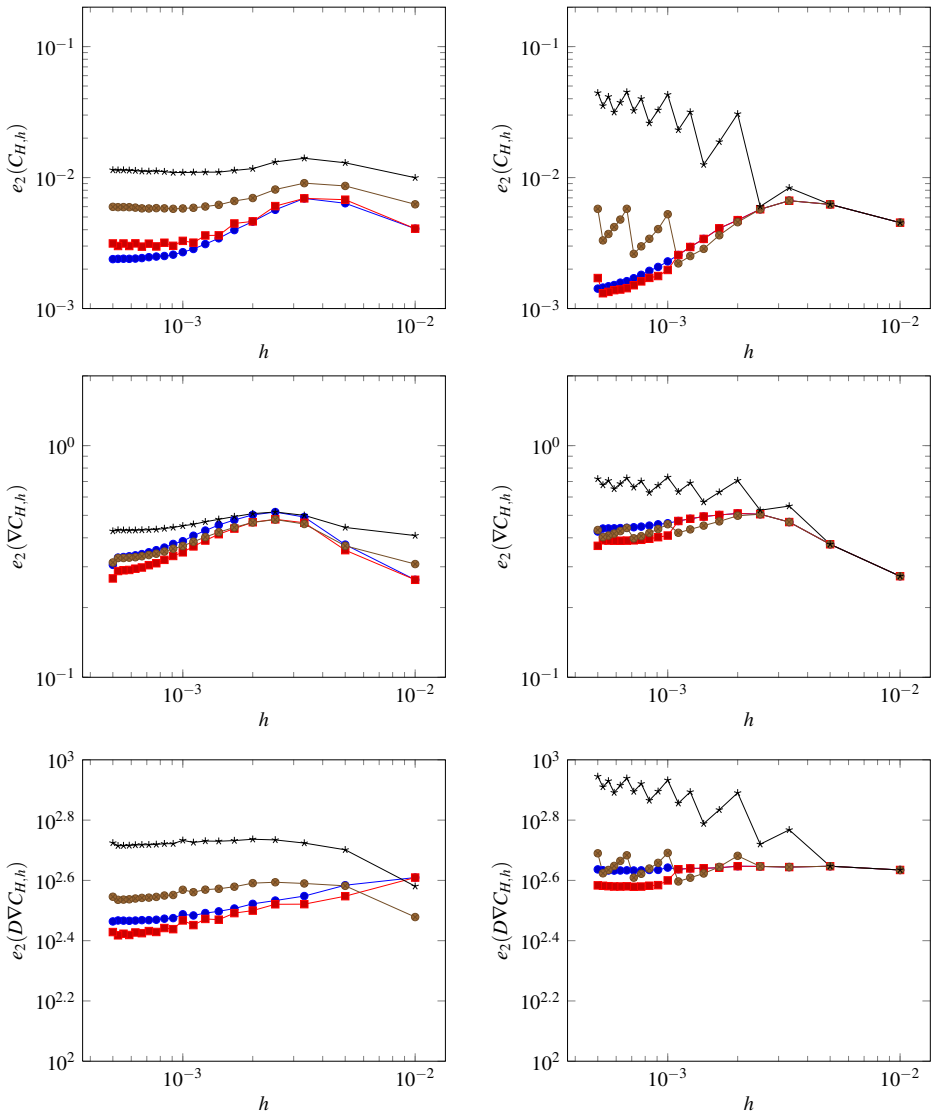


FIGURE 4.16 – Exemple 4.10 : $e_2(C_{H,h})$ (haut), $e_2(\nabla C_{H,h})$ (milieu) et $e_2(D\nabla C_{H,h})$ (bas) pour $H = 0.1$ (gauche) et $H = 0.02$ (droite). Sur chaque courbe, on a tracé l'évolution de l'erreur en fonction de h pour quatre valeurs de ρ : 0 (—●—), 0.05 (—■—), 0.10 (—●—) et 0.20 (—⋆—).

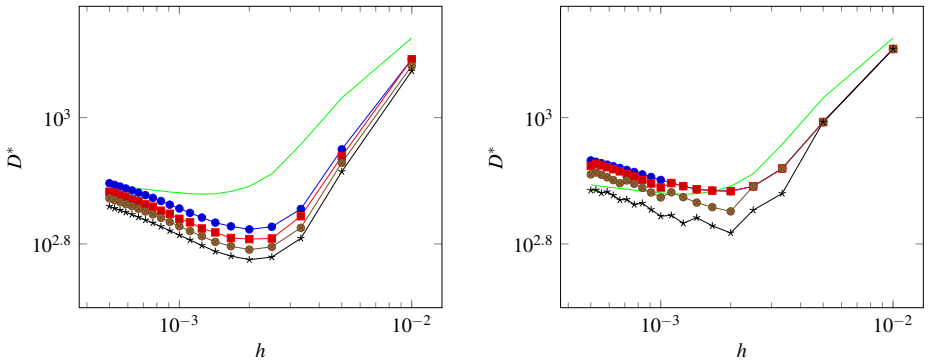


FIGURE 4.17 – Exemple 4.10 : évolution de la diffusivité équivalente D^* pour $H = 0.1$ (gauche) et $H = 0.02$ (droite). Sur chaque courbe, on a tracé l'évolution de D^* en fonction de h pour quatre valeurs de ρ : 0 (—●—), 0.05 (—■—), 0.10 (—●—) et 0.20 (—⋆—). La courbe verte correspond au coefficient calculé à partir de la solution directe.

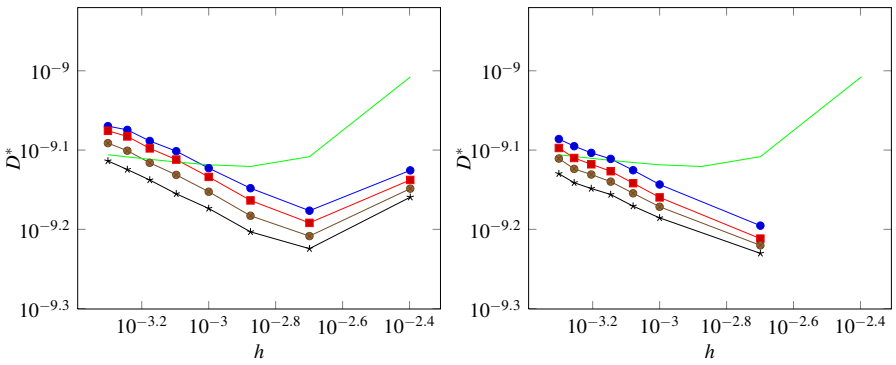


FIGURE 4.18 – Exemple 4.10 : évolution de la diffusivité équivalente D^* pour $h = 0.02H$ (gauche) et $h = 0.01H$ (droite). Sur chaque courbe, on a tracé l'évolution de D^* en fonction de h pour quatre valeurs de ρ : 0 (—●—), 0.05 (—■—), 0.10 (—●—) et 0.20 (—⋆—). La courbe verte correspond au coefficient calculé à partir de la solution directe.

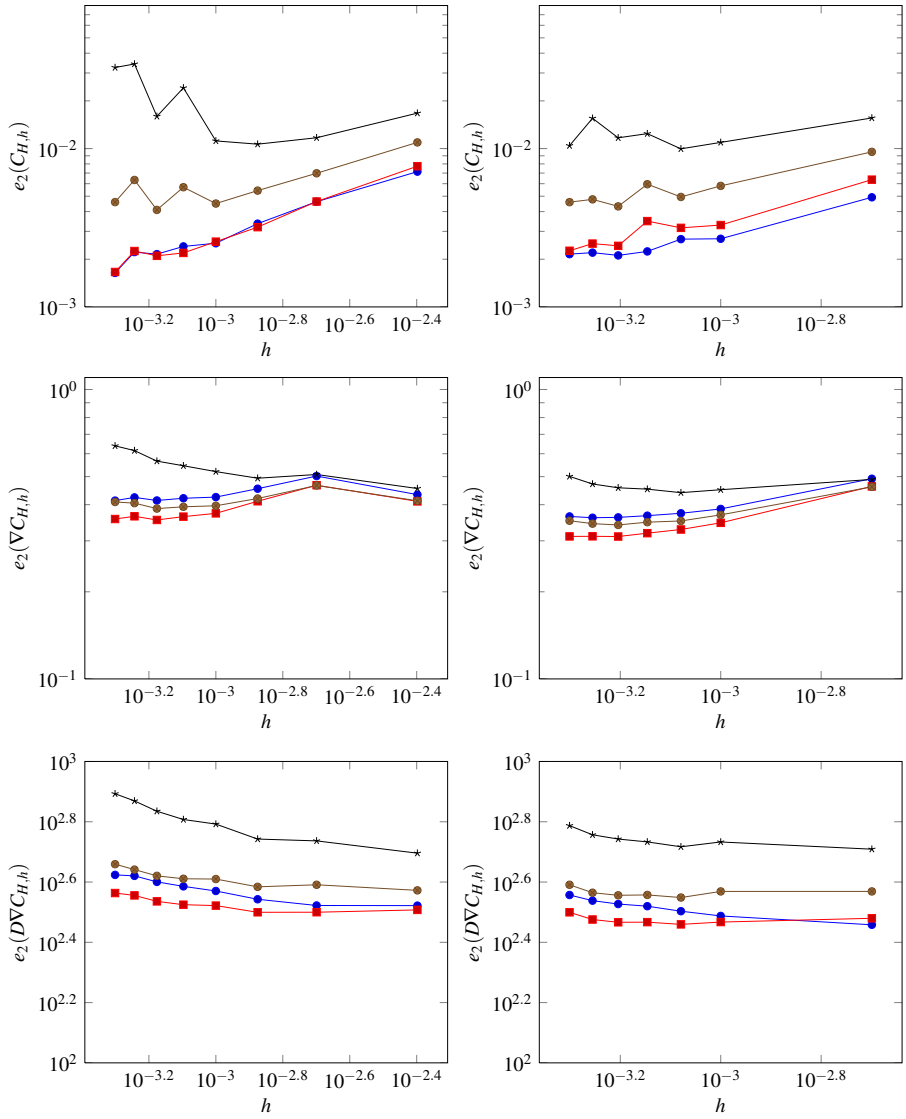


FIGURE 4.19 – Exemple 4.10 : erreurs $e_2(C_{H,h})$ (haut), $e_2(\nabla C_{H,h})$ (milieu) et $e_2(D\nabla C_{H,h})$ (bas) pour $h = 0.02H$ (gauche) et $h = 0.01H$ (droite). Sur chaque courbe, on a tracé l'évolution de l'erreur en fonction de h pour quatre valeurs de ρ : 0 (—●—), 0.05 (—■—), 0.10 (—●—) et 0.20 (—⋆—).

4.3.3.5 Évolution de la diffusivité équivalente du milieu.

Les diffusivités équivalentes D^* issues des solutions calculées par la méthode MsFEM sont présentées à la Figure 4.17 (évolution en fonction de h à H fixé) et à la Figure 4.18 (évolution en fonction de h à $\alpha = h/H$ fixé). On note que la diffusivité suit la même évolution quelle que soit la valeur choisie pour le taux de sur-échantillonnage ρ. Par contre, si à la Figure 4.17 cette évolution correspond bien à celle des valeurs obtenues par résolution directe (en vert sur les courbes), ce n'est pas le cas de la Figure 4.18. On ne peut donc réellement considérer qu'il y a convergence de la diffusivité équivalente, même si, au final, l'écart entre les valeurs calculées ne dépasse pas 10%.

Cet écart est sans doute dû aux erreurs dans l'évaluation du flux évoquées dans les paragraphes précédents.Le coefficient D^* étant en effet calculé à partir du flux, il est logique qu'une erreur sur l'un se reporte sur l'autre.

4.3.3.6 Repartition locale de l'erreur.

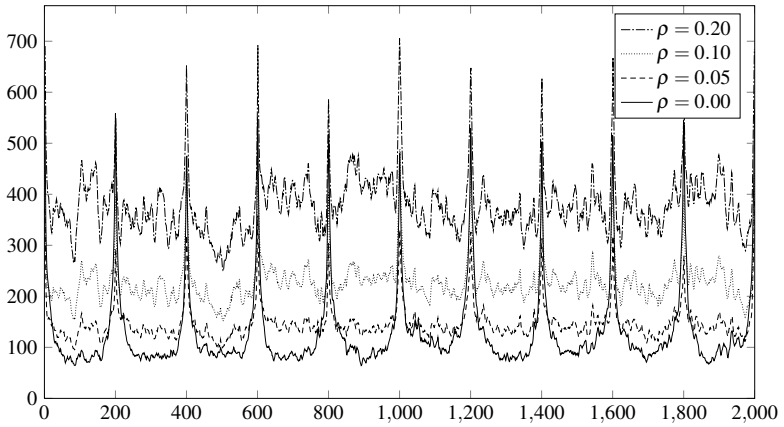


FIGURE 4.20 – Coupe de la densité de flux à travers le domaine Ω : différence entre la solution méthode multi-échelle $D_h \nabla C_{H,h} \cdot \mathbf{n}$ pour $(H,h) = (0.1, 5.10^{-4})$, et la solution de référence $D_r \nabla C_r \cdot \mathbf{n}$.

La Figure 4.20 présente une coupe de l'erreur, sur le flux $D\nabla C$, entre le cas $(H,h) = (0.1, 5.10^{-4})$ et la solution de référence. On note que l'erreur est trois à quatre fois plus importante le long des frontières de macro-éléments.

Ce phénomène de *couche limite*, décrit dans [90], devrait normalement être limité par le sur-échantillonnage, ce qui n'est pas le cas ici. Choisir $\rho = 0.05$ permet bien de réduire les valeurs maximales de l'erreur d'environ 30%, mais augmente aussi la valeur minimale de l'erreur, si bien, qu'en moyenne, l'erreur augmente. Quand aux cas $\rho = 0.1$ et $\rho = 0.2$, ils donnent de moins bons résultats sur tous les tableaux.

Le phénomène de *couche limite* n'étant pas limité ici, il est compréhensible que l'erreur sur la densité de flux augmente avec le nombre de *frontières*, c'est-à-dire, en pratique, avec le nombre M de macro-éléments.

4.4 Conclusions.

Ce chapitre présente une implémentation autonome de la méthode multi-échelle, réalisée durant la première année du travail de thèse. Cette maquette Python, une fois validée, a été utilisée pour résoudre des problèmes de diffusion : un cas théorique et un autre, plus complexe, modélisant une pâte cimentaire $2D$. De ces résultats,on a pu mettre en lumière plusiers problèmes.

Tout d'abord, il apparait sur les cas théoriques que l'erreur de la méthode diminue avec H, c'est-à-dire quand le nombre de cellules augmente. Une plus grande division du domaine permet donc d'obtenir de meilleurs résultats. Même s'il ne s'agit pas à proprement parler d'une convergence de la méthode $Q_1/VF9$, comme expliqué à la section §4.2.1, c'est toutefois un bon résultat

La méthode de recouvrement [79] permet normalement de corriger certains défauts des méthodes multi-échelles. Ses avantages dans le cadre de l'homogénéisation théorique sont bien connus. En reprenant les notations de la section §4.2.1, on a :

$$\|C^\varepsilon - C^\varepsilon_{H,h}\|_2 \leq A_1 \left(H + \sqrt{\frac{\varepsilon}{H}} + \frac{h}{\varepsilon} \right) \qquad \text{si } \rho = 0, \tag{4.23}$$

$$\|C^\varepsilon - C^\varepsilon_{H,h}\|_2 \leq A_2 \left(H + \frac{\varepsilon}{H} + \frac{h}{\varepsilon} \right) \qquad \text{si } \rho > 0. \tag{4.24}$$

Le lien théorique entre la convergence de la méthode multi-échelle et la valeur du taux de sur-échantillonnage ρ est cependant mal connu, car on ne sait expliciter la dépendance de A_2 en ρ. Tout au plus sait-on qu'augmenter ρ au-delà d'un certain point ne fait pas ni converger ni diverger la méthode multi-échelle, la constante A_2 se stabilisant rapidement à une valeur constante non nulle.

En pratique cependant, lorsque H diminue, l'erreur à tendance à stagner pour le cas de la pâte cimentaire. Les forts sauts de diffusivité du milieu, dont l'ordre de grandeur peut atteindre 10^8, sont sûrement à l'origine de ces problèmes. Il semble difficile, voir impossible, de choisir optimalement les différents paramètres de la méthode dans un cas aussi raide. En conséquence de quoi, la méthode de recouvrement est ici inefficace. Pire un recouvrement trop important déteriore considérablement les résultats.

Plusieurs pistes ont été considérées pour tenter de régler ce problème.

Tout d'abord, on a augmenté la taille des sytèmes étudiés. Augmenter la précision de la discrétisation permet en effet une capture plus fine des effets de couche-

limites via la méthode de recouvrement. Comme les calculs présentés dans ce chapitre atteignent les capacités maximales d'un ordinateur séquentiel, il a été nécessaire de changer le cadre d'implémentation de la méthode Q_1/*VF9* (programme Python sur un ordinateur) pour une interface plus robuste. Le choix s'est porté sur le code *MPCube* [61], basé sur le noyau de calcul *Trio-U* [171]. Cette nouvelle implémentation est présentée au chapitre §5.

Profitant de ce changement d'implémentation, on a considérablement élargi le cadre de travail, l'ambition affichée étant de pouvoir traiter des matériaux cimentaires 2*D* et 3*D*. Afin de modéliser au mieux les géométries des matériaux cimentaires, la contrainte sur les maillages a été supprimée. Les maillages fins ne sont plus quadrangulaires réguliers, mais quelconques. Ils doivent donc être générés avant toute simulation. Le chapitre §6 contient les méthodes développées au sein de la plate-forme *SALOME* [163] afin de pouvoir discrétiser rapidement et automatiquement les domaines de travail, c'est-à-dire l'ensemble des cellules $\hat{K}$. Il décrit ainsi comment l'on obtient, à partir d'une description d'un matériau cimentaire, les maillages nécessaires aux simulations *MPCube*.

Afin de faire face à l'importante raideur des problèmes de diffusion au sein des matériaux cimentaires, on a remplacé, à l'échelle fine, la méthode des Volumes Finis *VF9* par la méthode des Volumes Finis *VFDiam*. Introduite par *Coudière* et al. [79], cette méthode est depuis utilisée par le CEA pour travailler sur des milieux à forts contrastes et/ou à fortes anisotropies comme les matériaux cimentaires. Elle est décrite à la section §3.2.3.

Il apparaît qu'une non-conformité trop importante des Éléments Finis pénalise les résultats de la méthode Q_1/*VF9*, en particulier au niveau de la résolution du problème grossier. Il semble donc judicieux de remplacer, à l'échelle grossière, la méthode d'Éléments Finis classique par une méthode plus adaptée aux fonctions non-conformes. Le choix s'est portée sur une méthode discontinue de Galerkin, plus précisement une méthode de *pénalité intérieure*, ou IP [29, 82]. Elle est décrite à la section §3.3.2.

De ces deux choix de méthodes, on a construit deux couplages grossier/fin : le couplage Éléments Finis et *VFDiam*, que l'on note Q_1/*VFDiam*, et le couplage Galerkin discontinu et *VFDiam*, noté GD/*VFDiam*. Les résultats obtenus avec ces deux couplages sont présentés au chapitre §7.

Troisième partie

Mise en œuvre en dimension 3.

Chapitre 5

Mise en œuvre dans le cas $3D$: simulations *MPCube*.

Introduction.

Ce chapitre est consacré à la chaîne de calcul multi-échelle *SALOME-MPCube*, développée dans le cadre de cette thèse. Cette chaîne regroupe tous les outils nécessaires à la réalisation de simulations numériques par les méthodes Q_1/*VFDiam* et GD/*VFDiam*. Une description précise de ces deux méthodes est disponible au chapitre §3. La chaîne de calcul utilise principalement deux outils : la plate-forme *SALOME* [163] et le code de calcul *MPCube* [61].

La plate-forme *SALOME*, présentée à la section §6.1, intervient tout d'abord au début de la chaîne de calcul, afin de construire les maillages nécessaires aux futurs calcul *MPCube*. Cette partie de la chaîne de calcul est l'objet du chapitre §6.

Le code *MPCube*, basé sur le cœur de calcul *Trio-U* [171], intervient dans un second temps. Il résout les problèmes locaux, puis assemble et résout le problème grossier. Ces fonctionnalités, liées aux méthodes multi-échelles, n'existent pas dans le code originel de *MPCube*. Elles ont donc fait l'objet de développements spécifiques.

En fin de chaîne, *SALOME* intervient à nouveau, par le biais de ses outils MED-SPLITTER et VISU. La plate-forme permet d'effectuer des opérations de postraitement sur les champs solutions : calcul de la solution fine $C_{H,h}$, extraction et fusion des champs solutions, etc.

La chaîne de calcul multi-échelle *SALOME-MPCube* a été utilisée pour résoudre différents problèmes de diffusion au sein de matériaux cimentaires, que l'on présente au chapitre §7.

Dans un premier temps, on décrit les étapes de la chaîne de calcul multi-échelle *SALOME-MPCube*. On s'attache en particulier à présenter les possibilités de parallélisme étape par étape.

La seconde partie du chapitre est consacrée aux détails techniques de l'implémentation dans le code *MPCube* de la chaîne de calcul. On y présente tout d'abord les codes de calculs *Trio-U* et *MPCube* (§5.2.1) avant de détailler les différents objets ajoutés au code *MPCube* originel par la méthode multi-échelle (§5.2.2 et §5.2.3). On présente en outre une série de simulations ayant permis de qualifier l'implémentation de ces objets.

5.1 La chaîne de calcul multi-échelle *SALOME-MPCube*.

On peut espérer disposer de deux types de parallélisme au sein de la chaîne de calcul multi-échelle *SALOME-MPCube*.

Il y a tout d'abord le parallélisme intrinsèque aux méthodes multi-échelles, dit *extra*-cellulaire. La grande majorité du travail à effectuer sur les cellules du domaine, par exemple leurs discrétisations et la résolution des problèmes locaux, peut en effet s'effectuer indépendamment d'une cellule à l'autre. On peut donc traiter plusieurs cellules de front.

Il est également possible d'utiliser un second type de parallélisme, dit *intra*-cellulaire. Il s'agit de profiter des fonctionnalités de *SALOME* et *MPCube* qui permettent d'assigner plusieurs processeurs à une unique tâche. *MPCube* dispose en effet de solveurs parallèles, qui permettent de répartir l'effort de calculs de gros et très gros problèmes sur plusieurs cœurs de calculs. De même, *SALOME* peut faire appel à des mailleurs parallèles.

La Figure 5.1 présente un organigramme simplifié de la chaîne de calcul *SALOME-MPCube*. Sont également représentés les degrés de parallélisme, théoriques et implémentés, disponibles pour chaque module de l'organigramme.

Il est à noter que les étapes les plus coûteuses ont été parallélisées : résolution des problèmes locaux, construction des fonctions de bases et calcul des matrices associées. Comme on le détaille un peu plus bas, la génération des maillages est partiellement parallélisée dans la mesure où on construit chaque maillage de cellule indépendamment des autres. Pour des raisons techniques, cette construction s'effectue cependant séquentiellement, c'est-à-dire une cellule après l'autre.

5.1.1 Génération des maillages.

La première étape de la chaîne de calcul *SALOME-MPCube* consiste à discrétiser l'ensemble des cellules du découpage multi-échelle. Plus précisement, il s'agit, partant d'une description de la géométrie du domaine, d'obtenir un jeu de maillages exploitables par le code de calcul *MPCube*. Cette tâche est dévolue à la plate-forme *SALOME* et le chapitre §6 décrit les fonctionnalités spécifiquement développées dans ce but.

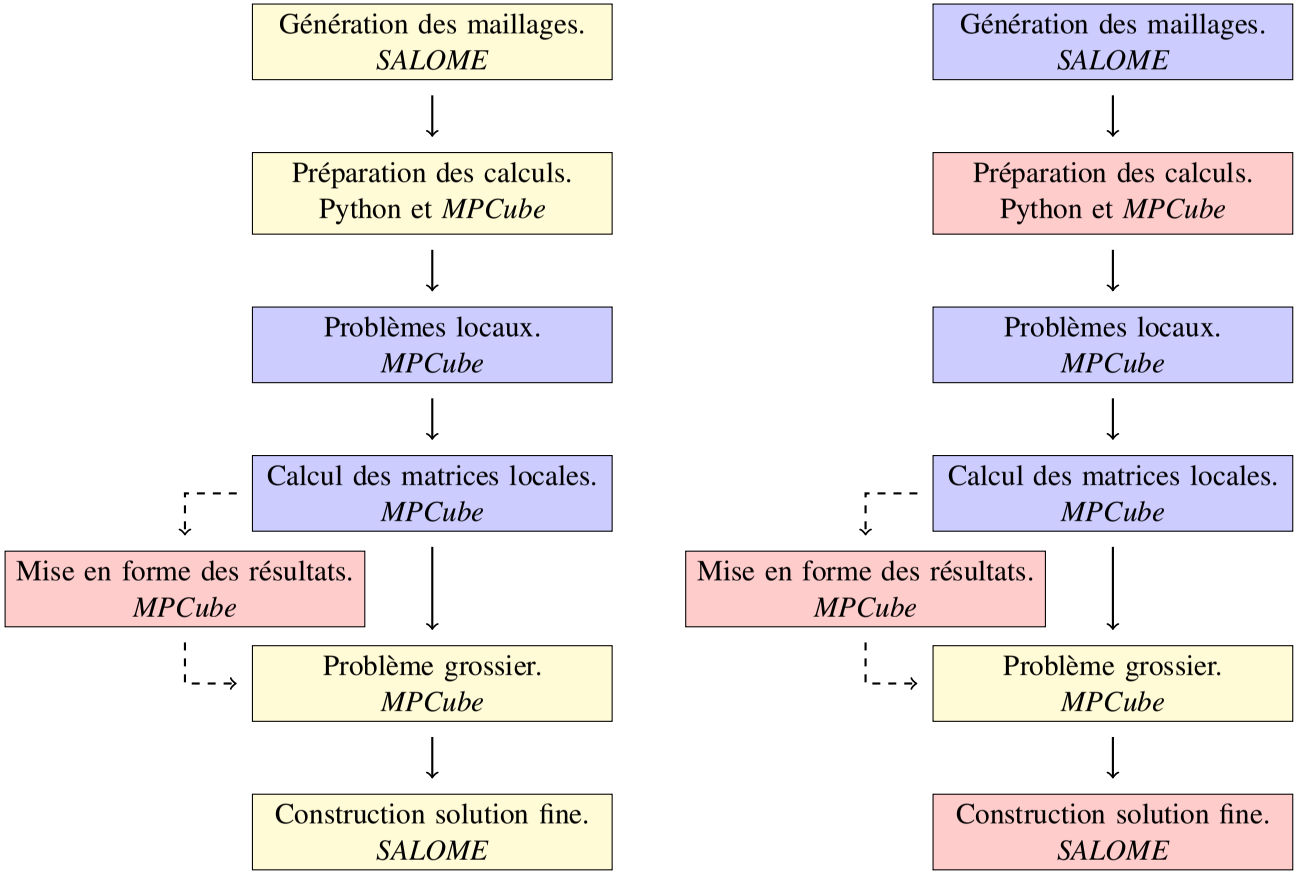


FIGURE 5.1 – Organigramme de la chaîne de calcul *SALOME-MPCube*. Les modules sont colorés en fonction du degré de parallélisme implémenté (à gauche) et théoriquement atteignable (à droite) : parallélismes *intra* et *extra*-cellulaire (en bleu), parallélisme *extra*-cellulaire seul (en rouge) et tâche séquentielle (en jaune).

Du point de vue du parallélisme, la génération de maillages dispose d'un parallélisme *extra*-cellulaire presque total. En effet, une fois les informations géométriques attribuées aux cellules, c'est-à-dire une fois les inclusions réparties (cf. Alg. 6.7 L.1), les travaux de construction géométrique et de discrétisation sont indépendants d'une cellule à l'autre. Dans le cas de la construction de maillages coïncidents pour la méthode GD/*VFDiam*, décrite à la section §6.3, il faut cependant tenir compte de la construction du maillage de référence (cf. Alg. 6.9 L.2), une tâche qui ne peut se faire que séquentiellement.

L'implémentation de ce module a été réalisée de manière à profiter de ce parallélisme. Ainsi, la fonction discrétisant une cellule ne requiert que la liste des inclusions correspondantes en entrée, et est appelée une fois par cellule. L'implémentation est donc indépendante d'une cellule à l'autre. Malgré cela, en pratique la discrétisation des cellules se fait séquentiellement. En effet, travailler de front sur plusieurs cellules demande de manipuler plusieurs instances de *SALOME*, ce qui est techniquement complexe car de nombreuses ressources sont partagées. Par le biais du module YACS, il est théoriquement possible de lancer une seule instance de *SALOME* qui piloterait les constructions paralléles. Cependant, cette option n'a pas été mise en place, le temps manquant pour maîtriser ce module.

La génération des maillages dispose d'un parallélisme *intra*-cellulaire. En effet, si la construction géométrique de la cellule ne peut que se faire séquentiellement, il n'en est pas de même de la discrétisation du domaine. *SALOME* dispose en effet d'un mailleur parallèle GHS3D PARALLEL [103] pouvant tirer partie de la disponibilité de plusieurs processeurs.

L'utilité de cette option reste cependant confinée à la création de très gros maillages, dépassant la dizaine de millions de volumes. Au contraire, le principe premier des méthodes multi-échelles est de travailler sur une série de problèmes de tailles raisonnables. Cette forme de parallélisme n'a donc pas été mise en œuvre.

5.1.2 Préparation des calculs.

Une fois les maillages obtenus, la chaîne de calcul *SALOME-MPCube* prépare les fichiers nécessaires aux simulations *MPCube*.

Le code de calcul *MPCube*, tout comme le code *Trio-U* sur lequel il est basé, fonctionne grâce à des jeux de données, nommés fichiers *data*. Ces fichiers définissent les caractéristiques des simulations à résoudre : termes sources, conditions aux limites, schéma Volumes Finis utilisé, solveurs, etc. Pour chaque cellule du domaine, un dossier de travail est construit, et un script Python génère ensuite automatiquement le fichier *data* correspondant aux problèmes de cellules et au calcul des matrices locales. Un exemple commenté de jeu de données est disponible à l'annexe §B.2.1.

L'appel à la commande MAKE_PAR.DATA achève la préparation de la simulation. Cette commande native de *Trio-U* construit les fichiers nécessaires à la réso-

lution parallèle du problème. En particulier, elle répartit le domaine de travail, ici le maillage d'une cellule, sur un nombre donné de processeurs et adapte les fichiers *data* en conséquences.

La Figure 5.2 présente un état du dossier de travail consacré à la cellule $(0,0,0)$, une fois terminée la préparation de la simulation.

```
1   ls work/x0/y0/z0:

    CELLULE3D_0_0_0_0000.Zones          Eléments liés au processeur $0$.
    CELLULE3D_0_0_0_0001.Zones          Eléments liés au processeur $1$.
    DEC_data_cellule3D_0_0_0.data       La répartition du milieu sur les processeurs.
6   DEC_data_cellule3D_0_0_0.err
    DEC_data_cellule3D_0_0_0.log
    DEC_data_cellule3D_0_0_0.out
    IntFond.file                        Fichiers décrivant les zones (fond, inclusions)
    IntFond.ssz                         du milieu.
11  IntInclusionD0.file                 Il y a un fichier .file et .ssz par zone.
    IntInclusionD0.ssz
    IntInclusionD1.file
    IntInclusionD1.ssz
    PAR_data_cellule3D_0_0_0.data       Jeu de données pour la résolution parallèle.
16  SEQ_data_cellule3D_0_0_0.data       Jeu de données pour la résolution séquentielle.
    cellule3D_0_0_0.geom                Réecriture du maillage au format geom.
    cellule3D_0_0_0.med                 Maillage original.
    data_cellule3D_0_0_0.data           Jeu de données original.
    ssz.geo                             Paramètres Trio-U.
21  ssz_par.geo                         Paramètres Trio-U.
```

FIGURE 5.2 – Liste des fichiers nécessaires à une simulation *MPCube*. Le domaine *cellule3D_0_0_0*, qui contient trois zones de diffusivités (le fond et deux types d'inclusions), a été préparé pour une résolution parallèle sur 2 processeurs.

La préparation des fichiers de simulation est une tâche indépendante d'une cellule à l'autre. On dispose donc théoriquement d'un parallélisme *extra*-cellulaire. En l'état actuel des choses, celui-ci n'a pas été implémentée, et les cellules sont préparées les unes après les autres.

La commande MAKE_PAR.DATA est séquentielle, son rôle étant justement de préparer un futur travail parallèle. Il n'y a donc pas de parallélisme *intra*-cellulaire possible pour cette étape de la chaîne de calcul.

5.1.3 Résolution des problèmes de cellules.

La résolution des problèmes de cellules s'effectue par la méthode des Volumes Finis *VFDiam*, *via* le code de calcul *MPCube*. En effet, une fois les fichiers *data* correctement préparés, les problèmes de cellules deviennent des problèmes de diffusion classiques. Il n'est donc pas nécessaire d'implémenter de nouvelles fonctionnalités au code *MPCube*.

On dispose donc d'un code de calcul parallèle performant, dont l'implémentation a été qualifiée dans de précédents travaux [61]. D'un point de vue technique, on choisit d'utiliser le solveur BICGSTAB et le préconditionneur multi-grille BOOMERAMG [113] par le biais de la librairie PETSC [154]. Ces solveurs se sont révélés adaptés à la résolution de problèmes de diffusions au sein des matériaux cimentaires [70].

Sur une cellule donnée, on résout les problèmes de cellules les uns après les autres, c'est-à-dire que l'on détermine $\Psi^1_{\hat{K}}$, puis $\Psi^2_{\hat{K}}$, et ainsi de suite. On profite alors du parallélisme *intra*-cellulaire du code de calcul *MPCube* en faisant appel à des solveurs parallèles pour résoudre les problèmes de cellule. En assignant plusieurs processeurs à cette tâche, on s'assure ainsi que chaque problème est résolu efficacement.

Les problèmes de cellules étant indépendants d'une cellule à l'autre, on dispose ici d'un parallélisme *extra*-cellulaire total. Afin de l'exploiter, on fait appel à la fonction *Trio-U* native **Execute_parallel**. Cette fonction permet d'exécuter plusieurs instances de *Trio-U*, et donc de *MPCube*, en parallèle, tout en précisant le nombre de processeurs à allouer à chaque instance.

Ainsi la Figure 5.3 présente un jeu de données permettant de lancer la résolution des problèmes sur trois cellules différentes, chaque cellule se voyant attribuer deux processeurs. Il est possible d'assigner un nombre différent de cœurs de calculs aux différentes instances, ce qui peut être utile si l'une des cellules a une discrétisation plus fine par exemple.

```
Execute_parallel
{
  liste_cas 3 ./Travail/work/x0/y0/z0/PAR_data_cellule3D_0_0_0
              ./Travail/work/x1/y0/z0/PAR_data_cellule3D_1_0_0
                 ./Travail/work/x2/y0/z0/PAR_data_cellule3D_2_0_0
  nb_procs 3 2 2 2
}

FIN
```

FIGURE 5.3 – En exécutant *MPCube* avec ce jeu de données, on lance trois nouvelles instances de *MPCube* qui traiteront les problèmes d'une cellule chacune. On a alloué deux cœurs de calcul à chaque instance.

5.1.4 Construction de la base locale et calcul des matrices locales.

Comme décrit à la section §3.3.1, les solutions $(\Psi^l_{\hat{K}})_{1\leq l\leq n}$ des problèmes de cellules ne forment pas une base de l'espace V_H des solutions du problème grossier (3.2). On construit donc, sur chaque cellule, la base locale de Galerkin $(\Phi^l_K)_{1\leq l\leq n}$ par combinaison linéaire des $(\Psi^l_{\hat{K}})_{1\leq l\leq n}$ (3.25). C'est à partir de ces bases locales que l'on construit une base de Galerkin lors de la résolution du problème grossier.

Une fois les fonctions $(\Phi^l_K)_{1\leq l\leq n}$ obtenues, on calcule les matrices locales de masse, de raideur, de saut, etc. Lors de la résolution du problème grossier, ces matrices sont assemblées, l'assemblage dépendant du schéma choisi pour résoudre le problème grossier : méthode des Éléments Finis (cf. §3.3.1) ou méthode de Galerkin discontinue (cf. §3.3.2).

Le calcul des fonctions $(\Phi^l_K)_{1\leq l\leq n}$ et des matrices locales associées est réalisé par le biais de *MPCube*. La Figure 5.4 présente la liste des instructions *MPCube* utilisées pour effectuer ce travail. Ces commandes doivent être exécutées à la suite de

la résolution des problèmes de cellules, qui sont notés `probleme0` à `probleme7` à la Figure 5.4.

Il ne s'agit pas d'une fonctionnalité native du code de calcul, mais de développements réalisés au cours de ces travaux de thèse. On a développé une série d'objets qui couvre toutes les opérations informatiques nécessaires à l'assemblage du problème grossier. Cela comprend la restriction au macroélément K des fonctions définies sur la cellule $\hat{K}$, l'interpolation des solutions pour obtenir des valeurs sur les faces et sommets du maillage fin $T_h(\hat{K})$, etc. Ces développements sont présentés en détail à la section §5.2.2.

```
Coloration_coordonnee Coloration_macroelement
Lire Coloration_macroelement
{
  expression (x_ge_(0.000000))_and_(x_le_(1.000000))
        _and_(y_ge_(0.000000))_and_(y_le_(1.000000))
        _and_(z_ge_(0.000000))_and_(z_le_(1.000000))
  probleme_associe probleme0
}

Base_locale_DG1HW Phi_K
Lire Phi_K
{
  valeur_rho 0.0
  prefixe cellule3D_0_0_0_
  coloration Coloration_macroelement
  type_sauvegarde ALL
  sommets_macroelement 2 8 3 24
      0.0 0.0 0.0
      1.0 0.0 0.0
      0.0 1.0 0.0
      1.0 1.0 0.0
      0.0 0.0 1.0
      1.0 0.0 1.0
      0.0 1.0 1.0
      1.0 1.0 1.0
  gestionnaire
  {
    dossier_champs /home/simulation/Bases_locales/
    dossier_matrices /home/simulation/Matrices_locales/
    type_sauvegarde ALL
    liste_problemes_locaux
    {
      probleme0 ,
      probleme1 ,
      probleme2 ,
      probleme3 ,
      probleme4 ,
      probleme5 ,
      probleme6 ,
      probleme7
    }
  }
}

Construire_base_locale Phi_K
Calculer_matrices_EF Phi_K
```

FIGURE 5.4 – Commandes *MPCube* permettant la construction de la base locale $(\Phi_K^l)_{1\leq l\leq n}$ et le calcul des matrices locales associées. Ces commandes doivent être exécutées à la suite de la résolution des problèmes de cellules, notés `probleme0`, `probleme1`, etc.

La construction de la base locale et le calcul des matrices locales de *raideur* $\mathbb{K}_K$ (3.38), de *masse* $\mathbb{M}_K$ (3.39) et de *masse de bord* $\mathbb{M}_{b,F}$ (3.41) sont indépendants

d'une cellule à l'autre. On dispose donc pour ces opérations d'un parallélisme *extra*-cellulaire total.

Cependant, si l'on souhaite utiliser une méthode discontinue de Galerkin pour résoudre le problème grossier, on doit également calculer les matrices de *saut* $\mathbb{T}$ (3.47), de *pénalisation* $\mathbb{Y}$ (3.48), de *saut Dirichlet* $\mathbb{T}_b$ (3.49), et de *pénalisation Dirichlet* $\mathbb{Y}_b$ (3.50). Les termes de ces matrices sont les résultats de calculs d'intégrales sur les faces F du maillage grossier $T_H(\Omega)$. Ces calculs font intervenir les valeurs des fonctions de bases sur les deux éléments K^+ et K^- ayant F pour frontières. Les cellules correspondantes $\hat{K}^+$ et $\hat{K}^-$ ne sont donc pas indépendantes.

Pour conserver un parallélisme *extra*-cellulaire complet, on calcule les valeurs des fonctions élémentaires sur les faces de chaque macroélément, puis on stocke ces valeurs. Le calcul proprement dit des matrices de saut et de pénalisation est réalisé séquentiellement lors de l'assemblage du problème grossier.

Pour des raisons techniques, l'objet *MPCube* GESTIONNAIRE_PROJET qui apparaît à la Figure 5.4 nécessite d'avoir en mémoire les objets *MPCube* correspondants aux problèmes de cellules. Pour cette raison, cette étape de la chaîne de calcul doit être réalisée au sein de la même instance *MPCube* que les résolutions des problèmes de cellules. Les choix concernant le parallélisme de cette étape sont donc effectués à l'étape précédente de la chaîne de calcul (cf. §5.1.3). En particulier, la possibilité que plusieurs cœurs de calculs soient affectés à une même cellule (parallélisme *intra*-cellulaire) doit être pris en compte.

Remarque 5.1 *Les besoins de l'objet* GESTIONNAIRE_PROJET *forment une raison supplémentaire de ne pas calculer les termes des matrices de* sauts *et de* pénalisations *à cette étape de la chaîne de calcul. En effet, si on souhaite les calculer maintenant, il est nécessaire de réarranger le travail sur les cellules, non plus macroélément par macroélément, mais face par face. Pour chaque face, la méthode devra alors résoudre les problèmes des deux cellules correspondantes avant de pouvoir calculer les termes matriciels. Au final, chaque problème de cellule aura été résolu quatre fois en dimension deux et six fois en dimension trois. Face à ce gaspillage colossal de ressources, l'idée a donc été abandonnée.*

5.1.5 Mise en forme des résultats locaux.

La résolution des problèmes de cellules à l'étape précédente de la chaîne de calcul *SALOME-MPCube* conduit à la création de nombreux fichiers. Outre les fichiers temporaires créés par *MPCube*, on trouve les enregistrements de la base locale $(\Phi_K^l)_{1\leq l\leq n}$, ainsi que les matrices locales correspondantes. Il est également possible de stocker le flux de la base locale $(D\cdot\nabla\Phi_K^l)_{1\leq l\leq n}$.

La base locale $(\Phi_K^l)_{1\leq l\leq n}$ est sauvegardée au format LATA. Il s'agit du format le plus simple à manipuler au sein du code de calcul *MPCube*, ce qui en a fait le format privilégié des outils *MPCube* développés durant ces travaux de thèse. Le format LATA sauvegarde cependant chaque fonction Φ_K^l dans un fichier indépendant, ce

qui entraîne la création d'un grand nombre de fichiers lors de la résolution des problèmes locaux.

Afin d'économiser les ressources informatiques, on convertit les fichiers LATA au format MED. Ce format permet de conserver dans un seul fichier la discrétisation de la cellule $T_h(\hat{K})$ et les valeurs des fonctions $(\Phi_K^l)_{1\leq l\leq n}$ sur ce maillage. Cette conversion se déroule en deux étapes.

Tout d'abord, il y a la conversion proprement dite, c'est-à-dire la création d'un fichier MED à partir des fichiers LATA. Cette conversion est le fait d'une fonction native de *MPCube*, la méthode **Lata_To_Med**. Il suffit donc de lancer une instance de *MPCube* avec le jeu de données adéquat pour obtenir le résultat voulu. La Figure 5.5 présente un exemple d'un tel jeu de données.

Le fichier MED ainsi créé contient, outre les valeurs des fonctions $(\Phi_K^l)_{1\leq l\leq n}$, la discrétisation de la cellule $T_h(\hat{K})$. Cependant, la numérotation de cette discrétisation, sommets et volumes, n'est pas celle du maillage d'origine fournit par *SALOME*, mais une numérotation automatiquement construite par **Lata_To_Med**. On renumérote donc le maillage du fichier MED pour le faire correspondre à la numérotation originale. Cette renumérotation s'effectue par le biais de *MPCube*, le jeu de données correspondant étant reproduit à l'Annexe §B.2.2.

```
dimension 3

Lata_To_Med  /home/simulation/wX0/wY0/wZ0/cellule3D_0_0_0_Solution.lata
             /home/simulation/wX0/wY0/wZ0/cellule3D_0_0_0_Solution.med

FIN
```

FIGURE 5.5 – Commandes *MPCube* permettant la conversion d'un ensemble de fichiers LATA en un fichier MED.

La fonction **Lata_To_Med** ne fonctionne que séquentiellement : il n'est pas possible d'assigner plusieurs cœurs de calcul à cette tâche. De même, la renumérotation ne peut s'effectuer que séquentiellement, puisque le cœur de calcul a besoin de la discrétisation complète du milieu pour travailler, et non pas seulement d'une portion. Les problèmes *MPCube* de conversion et de renumérotation doivent donc tous deux être réalisés séquentiellement. La mise en forme des résultats ne dispose donc pas de parallélisme *intra*-cellulaire.

Cependant, cette mise en forme est indépendante d'une cellule à l'autre. On dispose donc d'un parallélisme *extra*-cellulaire complet, que l'on peut exploiter grâce à la fonction *MPCube* **Execute_parallel**. La Figure 5.6 présente un jeu de données permettant de mettre en forme simultanément les résultats de deux cellules. Ce jeu de donnée est très proche de celui permettant de résoudre les problèmes sur plusieurs cellules simultanément, présentés à la Figure 5.3. Contrairement à ce dernier cependant, on ne peut allouer ici qu'un seul cœur de calcul à chaque cellule.

```
dimension 3

Execute_parallel
{
  liste_cas 2 /home/simulation/wX0/wY0/wZ0/cellule3D_0_0_0_conversion
              /home/simulation/wX1/wY0/wZ0/cellule3D_1_0_0_conversion
  nb_procs 2 1 1
}

Execute_parallel
{
  liste_cas 2 /home/simulation/wX0/wY0/wZ0/cellule3D_0_0_0_projection
              /home/simulation/wX1/wY0/wZ0/cellule3D_1_0_0_projection
  nb_procs 2 1 1
}

FIN
```

FIGURE 5.6 – En exécutant *MPCube* avec ce jeu de données, on met en forme les résultats des cellules $(0,0,0)$ et $(1,0,0)$. On ne peut allouer ici qu'un seul cœur de calcul à chaque instance.

Remarque 5.2 *À ce stade de la chaîne de calcul* SALOME-MPCube*, tous les calculs préliminaires à la résolution du problème grossier sont terminés. On dispose non seulement d'une base de Galerkin, mais également des matrices locales, qui permettront d'assembler, puis de résoudre, le problème grossier (3.2). Ces calculs dépendent entre autres de la diffusivité D et du découpage choisi mais sont indépendants du problème grossier à résoudre. À domaine de travail* Ω *et maillage grossier* $T_H(\Omega)$ *fixés, on peut donc résoudre n'importe quel problème grossier.*

5.1.6 Assemblage et résolution du problème grossier.

Au sein de la chaîne de calcul *SALOME-MPCube*, deux méthodes sont disponibles pour résoudre le problème grossier (3.2) : la méthode des Éléments Finis, présentée à la section §3.3.1, et la méthode de Galerkin discontinue, décrite à la section §3.3.2. Une fois le problème matriciel assemblé et résolu, on dispose de C_H, la solution grossière du problème.

Aucune des méthodes grossières, Éléments Finis ou Galerkin discontinue, n'est disponible nativement dans le code *MPCube*, celui-ci se concentrant sur les méthodes Volumes Finis. Au cours de ces travaux de thèse, on a donc implémenté ces méthodes dans *MPCube*. Cette extension n'est pas indépendante mais conçue, au contraire, pour s'interfacer aux méthodes multi-échelles. Ainsi l'assemblage du problème matriciel fait appel aux matrices locales précédemment calculées. La résolution proprement dite du problème matriciel fait cependant appel aux solveurs disponibles dans *MPCube*.

La Figure 5.7 présente un jeu de donnés permettant de résoudre un problème grossier. L'implémentation des méthodes grossières et les exemples permettant de qualification sont détaillés à la section §5.2.3.

Par sa nature même, l'assemblage et la résolution du problème grossier ne demande aucun travail local. Les notions de parallélismes *intra*-cellulaire et *extra*-

```
dimension 3
Probleme_grossier_EFP1 mon_probleme_grossier
Lire mon_probleme_grossier
   {
    decoupage 3 5 5 5
    extremites 2 2 3 6 0.0 0.0 0.0 5.0 5.0 5.0
    dossier_travail /home/simulation/Travail/
    dossier_matrices_locales /home/simulation/Matrices_locales/
    terme_source 0
    conditions_limites
      {
         zm Dirichlet 1
         zp Dirichlet 0
         ym Neumann 0
         yp Neumann 0
         xm Neumann 0
         xp Neumann 0
      }
    solveur gen
       {
        seuil 1e-13
        impr
        solv_elem bicgstab precond ilu
            {
             type    2
             filling 10
            }
       }
   }
Resoudre_probleme_grossier mon_probleme_grossier
FIN
```

FIGURE 5.7 – Jeu de données permettant de résoudre un problème grossier. Les paramètres `decoupage` et `dossier_matrices_locales` permettent de lier le problème grossier au reste de la chaîne multi-échelle *SALOME-MPCube*.

cellulaire n'ont donc pas de ce sens à ce stade la chaîne de calcul.

Il est cependant possible d'assigner plusieurs cœurs de calculs à la résolution du problème grossier, et de profiter ainsi des capacités des solveurs parallèles de *MPCube*. Ce parallélisme s'apparente au parallélisme *intra*-cellulaire, au sens où plusieurs cœurs de calculs sont utilisés pour résoudre une tâche. Le parallélisme *extra*-cellulaire, au contraire, permet de résoudre de front plusieurs tâches distinctes.

En pratique cependant les problèmes grossiers sont de tailles réduites : ils ne dépassent que rarement la centaine de milliers d'inconnues. L'intérêt d'une résolution parallèle du problème grossier est donc limité et n'a pas été poursuivi.

5.1.7 Reconstruction de la solution fine.

La dernière étape de la chaîne *SALOME-MPCube* calcule au niveau fin la solution $C_{H,h}$, en pondérant les fonctions $(\Phi^I)_{I\in\mathscr{I}}$ par les valeurs de la solution grossière C_H (3.52).

Cette reconstruction n'intervient généralement que sur une portion restreinte du domaine. En effet, on résout ici des problèmes de *très grandes tailles* pour lesquels une résolution classique est techniquement irréalisable. Il est donc généralement impossible de sauvegarder, et à plus fortes raisons de visualiser, la solution fine $C_{H,h}$ d'un seul tenant, tant la somme des discrétisations fines $(T_h(K))_{K\in T_H(\Omega)}$ est immense. Il peut cependant être intéressant de reconstruire la solution sur une zone déterminée, par exemple autour d'une singularité. On ne manipule ainsi que des

blocs de tailles raisonnables, qui n'excèdent pas les capacités informatiques disponibles.

Sur le plan technique, la reconstruction de la solution fine se déroule en trois étapes. Dans un premier temps, le solution est calculée sur chaque macroélément K à partir des valeurs des fonctions $(\Phi_K^l)_{1\leq l\leq n}$, sauvegardées au format MED. On utilise pour cela les librairies SMESH et MEDMEM de la plate-forme *SALOME* qui permettent de manipuler les données contenus dans un fichier MED : visualisation de la discrétisation, opérations arithmétiques sur les champs de données solutions, etc.

En théorie, les fonctions $(\Phi_K^l)_{1\leq l\leq n}$ sont définies sur le macroélément K. En pratique cependant, elles sont stockées sur la même discrétisation que les fonctions $(\Psi_{\hat{K}}^l)_{1\leq l\leq n}$ qui ont servi à les calculer, c'est-à-dire sur la discrétisation fine $T_h(\hat{K})$ de la cellule. Il est donc nécessaire de restreindre les fonctions $(\Phi_K^l)_{1\leq l\leq n}$ à la seule discrétisation $T_h(K)$ du macroélément sous-jacent. On fait appel pour cela à MEDSPLITTER, un outil de la plate-forme *SALOME*. MEDSPLITTER permet, comme son nom l'indique, de diviser un maillage, et les champs associés, en plusieurs blocs.

Enfin on fusionne les solutions fines, maintenant disponibles macroélément par macroélément, en une seule discrétisation. Pour cela, on fait à nouveau appel à MEDSPLITTER. Il est en effet possible, avec le paramétrage adéquat, d'utiliser cette commande pour agréger des maillages. Schématiquement, les maillages, ainsi que les champs associés, sont placés côte à côte, puis renumérotés par blocs. MEDSPLITTER ne se soucie pas de relier les maillages entre eux, ni de fusionner des sommets ou des faces qui seraient confondues. Au terme de cette opération, on dispose cependant d'un unique fichier MED contenant le champ solution sur un patch de macroéléments.

5.2 Implémentation et qualification.

On détaille dans cette section l'implémentation de la chaîne de calcul multi-échelle au sein du code de calcul *MPCube*, en commençant par présenter *MPCube* et le noyau *Trio-U* (§5.2.1).

Compte tenu des outils disponibles dans la version native de *MPCube*, il n'est pas nécessaire de développer un outil spécifique pour résoudre les problèmes de cellules et on peut obtenir les fonctions $(\Psi_{\hat{K}}^l)_{1\leq l\leq n}$ par le biais des jeux de données appropriés.

La construction des bases locales $(\Phi_K^l)_{1\leq l\leq n}$ et le calcul des matrices locales associées demandent par contre un important travail d'implémentation, que l'on décrit à la section §5.2.2.

Enfin, on présente à la section §5.2.3 la structure encadrant la résolution des problèmes grossiers par les méthodes d'Éléments Finis et de Galerkin discontinue. Chaque partie comprend les résultats d'un ou plusieurs exemples de simulations qui permettent de qualifier les implémentations.

5.2.1 Le code de calcul multi-physique *MPCube*.

À l'origine du code de calcul *MPCube* se trouve le noyau *Trio-U* [171]. *Trio-U* est un logiciel de simulation numérique en mécanique des fluides (code CFD pour *Computational Fluid Dynamics*). Il est développé au Laboratoire de Modélisation et de Développement Logiciels (LMDL) de la Direction de l'Énergie Nucléaire (DEN) du Commissariat à l'Énergie Atomique et aux Énergies Alternatives (CEA).

Ce noyau est implémenté en C++ [80] selon une approche objet. Il propose un cadre de travail générique pour le développement de codes de calculs, notamment par la définition de notions élémentaires communes : problème, équation, opérateur, schéma en temps, discrétisation, etc. Le noyau *Trio-U* dispose d'une interface vers différentes bibliothèques de solveurs linéaires, telles que HYPRE [123] ou PETSC [154].

Trio-U fournit également les structures de données et les fonctionnalités nécessaires à la conception de codes de calculs parallèles. Il permet ainsi une gestion de la mémoire et des entrées sorties parallèles, des opérations distribuées sur les vecteurs et les matrices et des algorithmes de résolution distribuée des systèmes linéaires. Ces outils parallèles sont internes au code et prêts à l'emploi.

Outre le code de calcul *MPCube*, le noyau *Trio-U* est utilisé par un autre code du CEA/DEN : le code *Flica-Ovap* [130].

Le code de calcul multi-physique *MPCube* est développé au Laboratoire de Simulation des Écoulements et Transports (LSET) du CEA Saclay. Il a été initialement développé pour la simulation des écoulements multi-phasiques en milieux poreux mais de manière plus générale est adapté aux problèmes de diffusion-convection dans les milieux poreux : hydraulique, transport d'espèces, écoulements multi-phasiques, etc. Une importante phase de qualification a été récemment réalisée, notamment une vérification de l'ordre des schémas et une analyse de la scalabilité [19, 60, 62].

MPCube implémente notamment la méthode de Volumes Finis *VFDiam* [20]. Cette méthode, qui est détaillée à la section §3.2.3, a la particularité d'être utilisable en dimension deux et trois, et ce quelle que soit la forme des éléments de la discrétisation : triangle ou quadrangle en $2D$, tétraèdre ou héxahèdre en $3D$. Ce schéma est bien adapté au traitement des discontinuités des paramètres physiques. On l'utilise dans ces travaux de thèse pour résoudre les problèmes de diffusion au sein des matériaux cimentaires.

5.2.2 Construction des fonctions locales de Galerkin.

On présente maintenant les outils mis en place au sein du code de calcul *MPCube* afin de construire, sur chaque macroélément K du maillage grossier $T_H(\Omega)$, la base locale $(\Phi_K^l)_{1\leq l\leq n}$ à partir des solutions des problèmes de cellules $(\Psi_{\hat{K}}^l)_{1\leq l\leq n}$, puis de calculer les matrices locales associées.

Il s'agit de l'implémentation de l'étape §5.2.2 de la chaîne multi-échelle *SALOME-MPCube*. Elle a été réalisé en C++, et utilise le cadre de travail générique de *Trio-U*.

5.2.2.1 Présentation de l'objet COLORATION.

Le taux de sur-échantillonnage ρ est un paramètre important des méthodes multi-échelles développées au cour de ces travaux de thèse. Quand ρ est strictement positif, la chaîne de calcul doit manipuler deux domaines distincts : la cellule $\hat{K}$, sur laquelle sont définies les solutions des problèmes de cellules $(\Psi^l_{\hat{K}})_{1\leq l\leq n}$, et le macroélément K, sur lequel est définie la base locale $(\Phi^l_K)_{1\leq l\leq n}$. Par définition, le macroélément K est inclus dans la cellule $\hat{K}$.

On dispose d'une discrétisation fine $T_h(\hat{K})$ de la cellule $\hat{K}$, qui a permis la résolution numériques des problèmes de cellules. Afin de disposer, au sein de l'instance *MPCube*, d'une discrétisation $T_h(K)$ du macroélément K, on doit identifier les éléments k de $T_h(\hat{K})$ qui appartiennent à K. C'est le rôle de l'objet COLORATION.

L'objet COLORATION permet de *peindre* les volumes d'une discrétisation selon un critère donnée, dans le cas présent l'appartenance au macroélément K. Il est possible de peindre des volumes en fonction de leur groupe MED (cf. §6.2.2.4) mais aussi des coordonnées de leurs sommets ou de leur barycentre.

L'objet COLORATION permet également de construire la frontière de la zone peinte. Pour une face f de $T_h(\hat{K})$ si un, et un seul, des volumes auxquels appartient f est coloré, alors f appartient à la frontière de la zone peinte. Cette fonctionnalité permet d'identifier la discrétisation $T_h(\partial K)$ de la frontière ∂K du macroélément K.

Remarque 5.3 *Afin de pouvoir colorer le macroélément K dans la discrétisation $T_h(\hat{K})$ de la cellule $\hat{K}$, il est nécessaire que cette discrétisation soit* adaptée *au macroélément. On rappelle qu'un maillage est dit* adapté *si tout élément du maillage ; volume, face ou sommet ; appartient à un unique objet physique : le macroélément, la zone de recouvrement, une interface du domaine, etc. Cette contrainte est prise en compte lors de la génération des maillages de cellules, voir à ce propos le chapitre §6 et notamment la section §6.1.1.*

5.2.2.2 Présentation de l'objet GESTIONNAIRE_PROJET.

Lors d'une simulation *MPCube*, chaque problème de cellule est contenu dans un objet MPCUBE_PROBLEME_BASE indépendant des autres. On a développé l'objet GESTIONNAIRE_PROJET afin de pouvoir manipuler ces problèmes d'un seul tenant. Par le biais cet objet, on peut accéder aux valeurs des fonctions $(\Psi^l_{\hat{K}})_{1\leq l\leq n}$, ainsi qu'aux propriétés de la discrétisation de la cellule, comme par exemple les volumes des éléments.

De manière générale, le rôle de l'objet GESTIONNAIRE_PROJET est de faire le lien entre les problèmes de cellule et l'objet BASE_LOCALE qui, lui, construit

la base locale $(\Phi_K^l)_{1\leq l\leq n}$. Dans cette optique, GESTIONNAIRE_PROJET permet notamment d'interpoler les valeurs des fonctions $(\Psi_{\hat{K}}^l)_{1\leq l\leq n}$ aux faces et aux sommets du maillage, *via* les équations (3.23) et (3.24). Il permet également d'accéder aux valeurs des flux $(D\cdot\nabla\Psi_{\hat{K}}^l)_{1\leq l\leq n}$ calculées par le schéma Volumes Finis *VFDiam* selon l'approximation (3.22).

5.2.2.3 Calcul des fonctions $(\Phi_K^i)_{1\leq i\leq n}$ par l'objet BASE_LOCALE.

L'objet BASE_LOCALE permet de calculer les fonctions $(\Phi_K^i)_{1\leq i\leq n}$ sur chaque macroélément $K\in T_H(\Omega)$. Du point de vue informatique, il possède un objet GESTIONNAIRE_PROJET, qui lui donne accès aux problèmes de cellule et leurs solutions, et un objet *MPCube*Coloration, qui lui permet de distinguer le macroélément K au sein de la discrétisation $T_h(\hat{K})$ de $\hat{K}$.

On peut trouver à la Figure 5.4 le jeu de données *MPCube* permettant d'effectuer cette opération, tandis que l'Algorithme 5.1 résume les différentes étapes de travail.

Algorithme 5.1 Objet BASE_LOCALE : construction de la base locale $(\Phi_K^l)_{1\leq l\leq n}$.

1: Déterminer les sommets $(S_i)_{1\leq i\leq n}$ de K.
2: Interpoler les fonctions $(\Psi_{\hat{K}}^i)_{1\leq i\leq n}$ en les points $(S_i)_{1\leq i\leq n}$.
3: Résoudre le problème linéaire (5.1).
4: Calculer la base locale $(\Phi_K^i)_{1\leq i\leq n}$.
5: Sauvegarder la base locale au format LATA.

La base locale $(\Phi_K^i)_{1\leq i\leq n}$ est identique que l'on utilise une méthode d'Éléments Finis ou de Galerkin discontinue à l'échelle grossière. Les fonctions $(\Phi_K^i)_{1\leq i\leq n}$ sont combinaisons linéaires des fonctions $(\Psi_{\hat{K}}^i)_{1\leq i\leq n}$, les solutions des problèmes de cellules (3.6). On détermine les coefficients $a_{i,l}$ en imposant, pour tout $1\leq i\leq n$:

$$\forall 1\leq j\leq n \qquad \Phi_K^i(S_j)=\sum_{1\leq l\leq n} a_{i,l}\Psi_{\hat{K}}^l(S_j)=\delta_{i,j}, \tag{5.1}$$

où on a repris les notations de la section §3.2.3, i.e on a noté $(S_i)_{1\leq i\leq n}$ les sommets du macroélément K et δ le symbole de Kronecker. On fournit à *MPCube* les coordonnées des points $(S_i)_{1\leq i\leq n}$ par le biais du jeu de données.

Avant de résoudre le problème linéaire (5.1), il est nécessaire d'obtenir les valeurs des fonctions $(\Psi_{\hat{K}}^i)_{1\leq i\leq n}$ en chacun des sommets $(S_i)_{1\leq i\leq n}$. La méthode *VFDiam* étant une méthode Volumes Finis centrée, la résolution numérique des problèmes de cellule permet d'obtenir une valeur sur chaque volume k de $T_h(\hat{K})$, et non sur les sommets. Lors de l'assemblage du schéma, elle approxime cependant les valeurs aux sommets, utilisant pour cela une méthode des moindres carrés (3.23).

Sur le plan informatique, les problèmes de moindres carrés ont été résolus pendant la résolution des problèmes de cellules, puis conservés en mémoire au sein de l'objet MPCUBE_PROBLEME_BASE correspondant. Les coefficients permettant de calculer $\Psi^i_{\hat{K}}(S_j)$ en fonction des valeurs de $\Psi^i_{\hat{K}}$ sur les éléments voisins de S_j sont donc disponibles sans besoin d'un travail supplémentaire. On accède donc à ces coefficients par le biais de l'objet GESTIONNAIRE_PROJET, ce qui permet d'assembler le problème linéaire (5.1).

Le problème (5.1) assemblé et résolu, on calcule la base locale $(\Phi^i_K)_{1\leq i\leq n}$. Ces fonctions, bien que définies en théorie sur le macroélément K, sont calculées sur la cellule $\hat{K}$. En effet, au sein de *MPCube*, les manipulations des champs solutions (multiplication par un scalaire, addition) sont aisées si les champs sont définis sur un même domaine. Le passage de la cellule au macroélément s'effectue en pratique lors de l'étape de mise en forme des résultats, que l'on a décrite à la section §5.1.5. Les fonctions $(\Phi^i_K)_{1\leq i\leq n}$ sont ensuite sauvegardées au format LATA. Elles sont ensuite mises en formes au format MED pour être utilisées plus loin dans la chaîne de calcul *SALOME-MPCube*, lors du recalcul de la solution fine $C_{H,h}$ (cf. §5.1.7).

Remarque 5.4 *Contrairement au format* MED*, le format* LATA *stocke les informations à travers plusieurs fichiers. La sauvegarde de la base locale conduit ainsi à la création de $n+1$ fichiers : un pour le maillage $T_h(\hat{K})$ et un par champ solution Φ^i_K. Le format* LATA *étant de plus incompatible avec la plate-forme* SALOME*, il n'apparaît pas comme un choix judicieux de format de stockage.*

Il s'agit en fait d'un choix d'implémentation. La sauvegarde au format LATA *est en effet très facile à implémenter au sein d'un objet* MPCube*. Au contraire, le format* MED *qui aurait été ici le choix évident, est étroitement lié aux mécaniques et objets fondamentaux de* Trio-U*. Il est donc difficile de le manipuler hors du cadre précis pour lequel il a été implémenté à l'origine.*

5.2.2.4 Calcul des matrices locales par l'objet BASE_LOCALE.

Afin de résoudre le problème grossier par une méthode d'Éléments Finis, on doit calculer les différents termes des matrices locales de raideur $\mathbb{K}_K$ (3.38), de masse $\mathbb{M}_K$ (3.39) et de masse de bord $\mathbb{M}_{b,F}$ (3.41) pour un élément $K \in T_H(\Omega)$ et une face $F \in T_H(\partial\Omega)$ donnés. Ces quantités sont définies par :

$$\forall 1 \leq i,j \leq n \qquad \mathbb{K}_K(i,j) = \frac{1}{2} \sum_{f \in T_h(\partial K)} \left(\Phi_K^i(f) \int_f D\nabla\Phi_K^j \cdot \mathbf{n} + \Phi_K^j(f) \int_f D\nabla\Phi_K^i \cdot \mathbf{n} \right) \tag{5.2}$$

$$\forall 1 \leq i,j \leq n \qquad \mathbb{M}_K(i,j) = \sum_{k \in T_h(K)} |k| \Phi_K^i(k) \Phi_K^j(k) \tag{5.3}$$

$$\forall 1 \leq i,j \leq n \qquad \mathbb{M}_{b,F}(i,j) = \sum_{f \in T_h(F)} |f| \Phi_K^i(f) \Phi_K^j(f) \tag{5.4}$$

Une description précise de ces équations et un rappel des différentes notations est disponible à la section §3.3.1.

Algorithme 5.2 Objet BASE_LOCALE : calcul des matrices locales.

1: Coloration du macroélément K et de sa frontière.
2: Interpolation des fonctions $(\Phi_K^i)_{1 \leq i \leq n}$ aux faces de $T_h(\partial K)$.
3: Calculer le flux des fonctions $(\Phi_K^i)_{1 \leq i \leq n}$ aux faces de $T_h(\partial K)$.
4: Calculer les matrices locales.
5: Sauvegarder les matrices locales.
6: Sauvegarder les valeurs interpolées et le flux aux faces de $T_h(\partial K)$.

Ces matrices sont calculées par l'objet BASE_LOCALE en suivant les étapes de l'Algorithme 5.2. On commence par identifier les éléments k de $T_h(\hat{K})$ constituant une discrétisation $T_h(K)$ de K, ainsi que les faces f de $T_h(\hat{K})$ qui discrétisent $T_h(\partial K)$. Cette tâche est dévolue à l'objet COLORATION que l'on a décrit à la section §5.2.2.1.

Pour calculer les matrices locales de raideur $\mathbb{K}_K$ et de masse de bord $\mathbb{M}_{b,F}$, il est nécessaire d'obtenir les valeurs des fonctions $(\Phi_K^i)_{1 \leq i \leq n}$ en chacune des faces de $T_h(\partial K)$. La méthode *VFDiam* étant une méthode Volumes Finis centrée, la résolution numérique des problèmes de cellule permet d'obtenir une valeur sur chaque volume k de $T_h(\hat{K})$, et non sur les faces. Au cours de la construction théorique du schéma *VFDiam*, une valeur au barycentre G_f de la face f est calculée afin d'imposer l'égalité des flux à travers la face (3.21).

On utilise l'équation (3.21) pour interpoler les valeurs des fonctions $(\Phi_K^i)_{1 \leq i \leq n}$ aux faces $f \in T_h(\partial K)$. Les valeurs aux faces sont des combinaisons linéaires de valeurs aux éléments, que l'on connaît, et des valeurs aux sommets, dont on a expliqué l'interpolation à la section précédente §5.2.2.3.

Les coefficients de ces combinaisons linéaires font intervenir la diffusivité D du milieu et les propriétés géométriques (surface, orientation) des faces. Sur le

plan informatique, ils ont été calculés pendant la résolution des problèmes de cellules, puis conservés en mémoire au sein des objets MPCUBE_PROBLEME_BASE correspondants. On obtient donc ces coefficients par le biais de l'objet GESTIONNAIRE_PROJET sans besoin de travail supplémentaire.

La seconde quantité nécessaire au calcul de la matrice locale de raideur $\mathbb{K}_K$ est le flux des fonctions $(\Phi_K^i)_{1\leq i\leq n}$ à travers les faces f de $T_h(\partial K)$. Le flux des fonctions $(\Psi_{\hat{K}}^i)_{1\leq i\leq n}$ est calculé par le schéma *VFDiam* lors de la résolution des problèmes de cellules (cf. (3.22)). Par le biais de l'objet GESTIONNAIRE_PROJET, on récupère les valeurs de ces flux, puis on calcule le flux des fonctions $(\Phi_K^i)_{1\leq i\leq n}$ par sommation, les fonctions $(\Phi_K^i)_{1\leq i\leq n}$ étant combinaison linéaire des fonctions $(\Psi_{\hat{K}}^i)_{1\leq i\leq n}$.

Une fois que l'on dispose de toutes les quantités nécessaires, on calcule les matrices locales. On les sauvegarde ensuite afin de pouvoir les utiliser lors de la résolution du problème grossier.

Si on souhaite utiliser une méthode de Galerkin discontinue pour résoudre le problème grossier, on doit disposer des matrices de *saut* $\mathbb{T}_K$ (3.47), de *pénalisation* $\mathbb{Y}_K$ (3.48), de *saut Dirichlet* $\mathbb{T}_{b,F}$ (3.49), et de *pénalisation Dirichlet* $\mathbb{Y}_{b,F}$ (3.50). Une description précise de ces matrices est disponible à la section §3.3.2.3.

Comme on l'a expliqué à la section §5.1.4, et plus précisément à la Remarque 5.1, ces matrices ne sont pas calculées ici, mais lors de l'assemblage du problème grossier. En effet, la définition de ces matrices s'appuie sur le *saut* et la *moyenne* des fonctions $(\Phi_K^i)_{1\leq i\leq n}$ à travers les faces F de $T_H(\Omega)$. Le saut et la moyenne sur une face F, définis par la Définition 3.5, font appel aux valeurs issues des deux côtés de F, en l'occurence les valeurs des fonctions $(\Phi_K^i)_{1\leq i\leq n}$ sur deux macroéléments voisins.

Afin de pouvoir paralléliser la résolution des problèmes de cellules, chaque instance *MPCube* ne contient les informations relatives qu'à un unique macroélément. Il n'est donc pas possible d'accéder aux fonctions de bases d'un macroélément voisin. On ne peut par conséquent pas calculer les matrices locales $\mathbb{T}_K$, $\mathbb{Y}_K$, $\mathbb{T}_{b,F}$, $\mathbb{Y}_{b,F}$ à ce stade de la chaîne de calcul.

On sauvegarde donc les valeurs interpolées aux faces et le flux des fonctions $(\Phi_K^i)_{1\leq i\leq n}$ pour pouvoir calculer ces matrices lors de l'assemblage du problème grossier.

5.2.2.5 Qualification de l'objet BASE_LOCALE.

On valide dans cette section l'implémentation de l'objet BASE_LOCALE et des objets COLORATION et GESTIONNAIRE_PROJET.

Le calcul de la base locale $(\Phi_K^i)_{1\leq i\leq n}$ à partir des solutions des problèmes de cellules $(\Psi_{\hat{K}}^i)_{1\leq i\leq n}$ est simple à valider. Il suffit en effet de vérifier que les fonctions $(\Phi_K^i)_{1\leq i\leq n}$ valent bien 1 ou 0 sur les sommets $(S_l)_{1\leq l\leq n}$.

On se concentre donc sur le calcul des termes matriciels qui, outre les fonctions de bases $(\Phi_K^i)_{1\leq i\leq n}$, fait intervenir les formules d'interpolations aux sommets et aux faces des éléments et la coloration des éléments du maillage $T_h(\hat{K})$.

Exemple 5.1 *On considère dans un premier temps, le cas linéaire définit par :*

$$D = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix}$$
$$K =]0,1[\times]0,1[\times]0,1[$$
$$F = \{x = 0\}$$

où a, b et c sont des réels strictements positifs. Dans ce cas, les fonctions $(\Phi_K^i)_{1\leq i\leq n}$ sont explicitement connues, et ce pour toute valeur de ρ :

$$\Phi_K^1 = (1-x)(1-y)(1-z) \qquad \Phi_K^5 = (1-x)(1-y)z$$
$$\Phi_K^2 = x(1-y)(1-z) \qquad \Phi_K^6 = x(1-y)z$$
$$\Phi_K^3 = (1-x)y(1-z) \qquad \Phi_K^7 = (1-x)yz$$
$$\Phi_K^4 = xy(1-z) \qquad \Phi_K^8 = xyz$$

Les matrices $\mathbb{K}_K$, $\mathbb{M}_K$ et $\mathbb{M}_{b,F}$ sont symétriques. De plus, étant donnée la symétrie des fonctions $(\Phi_K^i)_{1\leq i\leq n}$, de nombreux termes de ces matrices sont identiques. On a ainsi :

$$\mathbb{M}_K(1,2) = \mathbb{M}_K(1,3) = \mathbb{M}_K(1,5),$$
$$\mathbb{M}_K(1,4) = \mathbb{M}_K(1,6) = \mathbb{M}_K(1,7),$$

des formules semblables existant pour les matrices $\mathbb{K}_K$ et $\mathbb{M}_{b,F}$. Les Figures 5.8, 5.9 et 5.10 présentent le maximum de l'erreur entre les valeurs théoriques ($\mathbb{K}_K$, $\mathbb{M}_K$ et $\mathbb{M}_{b,F}$) et numériques ($\tilde{\mathbb{K}_K}$, $\tilde{\mathbb{M}_K}$ et $\tilde{\mathbb{M}_{b,F}}$) des matrices locales :

$$e_\infty(\mathbb{A}) = \sup_{1\leq i,j\leq n} |\tilde{\mathbb{A}}(i,j) - \mathbb{A}(i,j)| \tag{5.5}$$

Il apparaît des simulations que les erreurs décroissent en $\mathcal{O}(h^2)$.

Exemple 5.2 *Le second exemple que l'on considère est le suivant :*

$$D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$
$$K =]0,1[^3$$
$$F = \{x = 0\}$$
$$\beta = \cos(\pi x)\cos(\pi y)\cosh(\pi z)$$

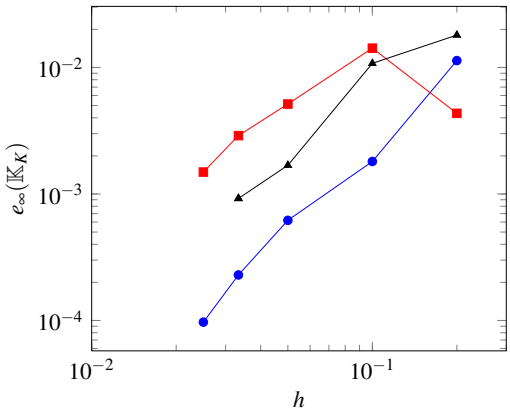


FIGURE 5.8 – Exemple 5.1 : maximum de l'erreur de calcul, en fonction de h, des coefficients de $\mathbb{K}_K$ pour $\rho = 0$ (—●—), $\rho = 0.05$ (—■—) et $\rho = 0.1$ (—▲—).

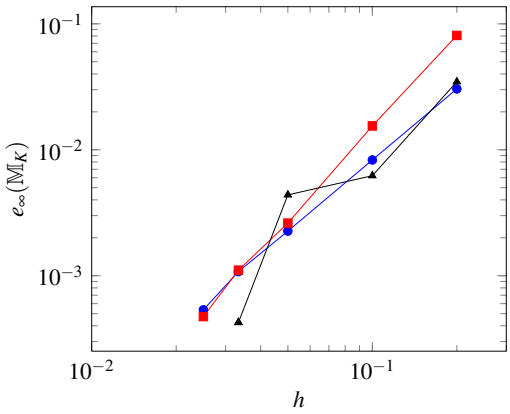


FIGURE 5.9 – Exemple 5.1 : maximum de l'erreur de calcul, en fonction de h, des coefficients de $\mathbb{M}_K$ pour $\rho = 0$ (—●—), $\rho = 0.05$ (—■—) et $\rho = 0.1$ (—▲—).

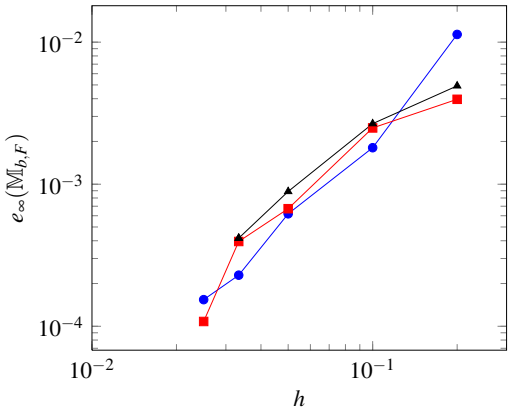


FIGURE 5.10 – Exemple 5.1 : maximum de l'erreur de calcul, en fonction de h, des coefficients de $\mathbb{M}_{b,F}$ pour $\rho = 0$ (—●—), $\rho = 0.05$ (—■—) et $\rho = 0.1$ (—▲—).

La solution du problème est alors :

$$\Psi = \cos(\pi x)\cos(\pi y)\cosh(\pi z)$$

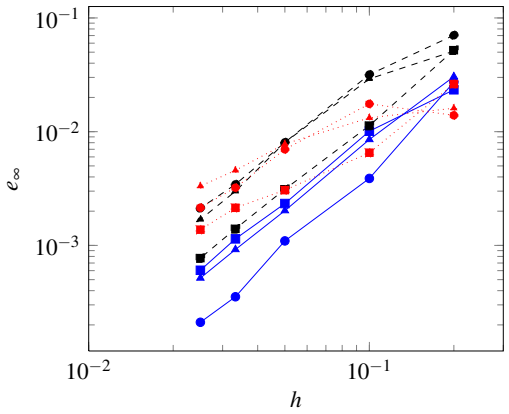


FIGURE 5.11 – Exemple 5.2 : erreur de calcul, en fonction de h, des coefficients $\mathbb{K}_K(\Psi,\Psi)$ (—●—), $\mathbb{M}_K(\Psi,\Psi)$ (—■—) et $\mathbb{M}_{b,F}(\Psi,\Psi)$ (—▲—) pour $\rho = 0$ (trait), $\rho = 0.5$ (pointillé) et $\rho = 0.1$ (tiret).

On peut alors calculer les termes (5.2)-(5.4) en posant $\Phi_K^i = \Phi_K^j = \Psi$. On présente à la Figure 5.11 l'erreur entre valeurs théoriques et numériques de ces trois termes pour différentes valeurs de ρ. Pour $\rho = 0$, les coefficients $\mathbb{K}(\Psi,\Psi)$ et $\mathbb{M}(\Psi,\Psi)$ et $\mathbb{M}_b(\Psi,\Psi)$ convergent en $\mathscr{O}(h^2)$. Quand le sur-échantillonnage est non nul, la convergence varie entre $\mathscr{O}(h^1)$ et $\mathscr{O}(h^2)$ selon les termes.

5.2.3 Résolution du problème grossier.

5.2.3.1 L'objet PROBLEME_GROSSIER.

Le code *MPCube* natif n'implémente aucune méthode d'Éléments Finis ou de Galerkin discontinue, y préférant les méthodes de Volumes Finis comme le schéma *VFDiam* par exemple. Même dans le cas contraire, il aurait été fort peu probable que ces méthodes soient compatibles avec la chaîne de calcul multi-échelle *SALOME-MPCube*. En effet, le principe même des méthodes multi-échelles est d'utiliser, pour résoudre le problème grossier, de fonctions issues des problèmes de cellules, et non d'une base usuelle comme les polynômes de Legendre.

Afin de pallier à ce manque, l'objet PROBLEME_GROSSIER a donc été ajouté au code *MPCube*. Il est déclinable en deux versions, respectivement les objets PROBLEME_GROSSIER_EFP1 et PROBLEME_GROSSIER_GAP1, qui permettent de résoudre un problème grossier dans un contexte multi-échelle par les méthodes d'Éléments Finis (cf. §3.3.1) et de Galerkin discontinue à pénalité intérieure (cf. §3.3.2) respectivement.

On trouve à la Figure 5.7 le jeu de données commandant la résolution d'un problème grossier par une méthode Éléments Finis. L'Algorithme 5.3 récapitule les étapes de cette résolution.

Algorithme 5.3 Objet PROBLEME_GROSSIER : résolution du problème grossier.

1: Assembler le problème grossier à partir des matrices locales.
2: Résoudre le problème grossier *via* un solveur de *MPCube*.
3: Sauvegarder la solution C_H.

Afin de résoudre le problème grossier, on assemble le problème matriciel correspondant à partir des matrices locales. Les matrices locales de raideur $\mathbb{K}_K$ (3.38), de masse $\mathbb{M}_K$ (3.39) et de masse de bord $\mathbb{M}_{b,F}$ (3.41) ont été calculées lors de la résolution des problèmes de cellules, un processus décrit à la section §5.2.2.4. L'implémentation de l'assemblage de la matrice globale ne pose pas de problème particulier, car il s'agit avant tout d'un réarrangement de termes qui a été détaillé aux sections §3.3.1 pour la méthode d'Éléments Finis et §3.3.2, pour la méthode Galerkin discontinue.

Les matrices locales spécifiques à la méthode de Galerkin discontinue, i.e les matrices de *saut* $\mathbb{T}_K$ (3.47), de *pénalisation* $\mathbb{Y}_K$ (3.48), de *saut Dirichlet* $\mathbb{T}_{b,F}$ (3.49), et de *pénalisation Dirichlet* $\mathbb{Y}_{b,F}$ (3.50), n'ont pas été calculées à lors de la résolution des problèmes de cellules. Les quantités nécessaires, valeurs et flux aux faces, ont cependant été stockées. Lors de la résolution du problème grossier, l'instance *MPCube* peut accéder à l'ensemble de ces quantités, tous macroéléments compris, et calculer les matrices.

Une fois le problème matriciel assemblé, il est résolu par le biais d'un des solveurs de *MPCube* et la solution grossière C_H est sauvegardée.

Remarque 5.5 *Le seul véritable lien entre le problème grossier et le code de calcul* MPCube *est l'utilisation d'un des solveurs de* MPCube *pour la résolution. On dispose d'une autre gamme de solveurs* via *la librairie* NUMERICAL PLATON *[167], utilisée au sein de la maquette Python autonome décrite au chapitre §4.*

Toute cette partie de la chaîne de calcul SALOME-MPCube *peut donc être implémentée au sein d'une maquette indépendante de* MPCube. *On a cependant préféré l'y intégrer pour des raisons de cohérence, afin de minimiser le nombre de codes de calcul nécessaires à la chaîne* SALOME-MPCube.

5.2.3.2 Qualification de l'implémentation.

On valide dans cette section l'implémentation des objets qui assemblent et résolvent le problème grossier, c'est-à-dire les objets PROBLEME_GROSSIER_EFP1 et PROBLEME_GROSSIER_GAP1. Il ne s'agit pas ici de tester les méthodes multi-échelles Q_1/*VFDiam* et GD/*VFDiam* dans leurs intégralités, mais d'assurer que les

méthodes d'Éléments Finis et de Galrkin discontinue ont été correctement implémentées.

Pour cela, on suppose que la diffusivité D est égale à la matrice *Identité* et que le recouvrement est nul. On retrouve ainsi le cadre de l'Exemple 4.4 avec $(a,b) = (1,1)$ pour la dimension 2 et celui de l'Exemple 5.1 avec $(a,b,c) = (1,1,1)$ pour la dimension 3. Les fonctions $(\Phi_K^i)_{1\leq i\leq n}$ sont alors explicitement connues sur chaque macroélément, et on peut déterminer précisément tous les termes matriciels nécessaires aux deux schémas grossiers.

On résout donc plusieurs exemples du problème de diffusion suivant :

$$\left\{\begin{array}{rcll} -\Delta C & = & f & \text{dans } \Omega, \\ C & = & g_D & \text{sur } \Gamma_D, \\ D\nabla C\cdot\mathbf{n} & = & g_N & \text{sur } \Gamma_N, \end{array}\right. \tag{5.6}$$

où (Γ_D,Γ_N) forme une partition de $\Gamma = \partial\Omega$ la frontière du domaine Ω. $\mathbf{n}$ designe le vecteur normal unitaire extérieure au domaine. f, g_D et g_N sont respectivement le terme source, la condition aux limites de Dirichlet et la condition aux limites de Neumann.

Dans tout ce qui suit, on suppose que $\Omega =]0,1[^d$, où d est la dimension de travail. La discrétisation $T_H(\Omega)$ de Ω se compose de macroéléments carrés ou cubiques de longueur H.

Exemples en dimension 2**.**

Exemple 5.3 *Afin de tester l'implementation des conditions aux limites de Dirichlet, on considère dans cet exemple que :*

$$\begin{aligned} f &= 0, \\ \Gamma_D &= \partial\Omega, \\ g_D &= \sin(\pi x)\sinh(\pi y). \end{aligned}$$

La solution du problème est alors :

$$C = \sin(\pi x)\sinh(\pi y).$$

Exemple 5.4 *On note $\Gamma_{y,m}$ et $\Gamma_{y,p}$ les faces du domaine Ω pour lesquelles on a $\{y=0\}$ et $\{y=1\}$ respectivement. Afin de tester l'implémentation des conditions aux limites de Neumann, on considère dans cet exemple que :*

$$\begin{aligned} f &= 0, \\ \Gamma_N &= \Gamma_{y,m}\sqcup\Gamma_{y,p}, \\ g_N &= \frac{\partial}{\partial\mathbf{n}}\left(\sin(\pi x)\sinh(\pi y)\right), \\ \Gamma_D &= \partial\Omega\setminus\Gamma_N, \\ g_D &= \sin(\pi x)\sinh(\pi y).. \end{aligned}$$

La solution du problème est alors :

$$C = \sin(\pi x)\sinh(\pi y)$$

Exemple 5.5 *Afin de tester l'implémentation du terme source, on considère dans cet exemple que :*

$$\begin{aligned} f &= 2(x(1-x)+y(1-y)),\\ \Gamma_D &= \partial\Omega,\\ g_D &= 0. \end{aligned}$$

La solution du problème est alors :

$$C = x(1-x)y(1-y).$$

Le critère de validation choisit ici est l'erreur normalisée maximale entre valeurs calculées et théoriques aux noeuds $I \in \mathscr{I}$:

$$e_\infty(C_H) = \max_{I\in\mathscr{I}} \frac{|C_H(I) - C(I)|}{1+|C(I)|} \tag{5.7}$$

Il ne s'agit pas du maximum de l'erreur entre solution théorique et calculée car on ne recalcule pas la solution $C_{H,h}$ au niveau fin. De cette manière, la discrétisation des macroéléments K n'influence le résultat que par le biais des matrices de raideurs et de masses, influence que l'on néglige dans cette partie.

La Figure 5.12 présente l'évolution de l'erreur $e_\infty(C_H)$ en fonction de H lors de la résolution par la méthode d'Éléments Finis (cercle) et de Galerkin discontinue (carré) des exemples précédents : Exemple 5.3 (—○— et —□—), Exemple 5.4 (—○— et —□—) et Exemple 5.5 (—○— et —□—).

Que ce soit pour la méthode d'Éléments Finis ou celle de Galerkin discontinue, tous les exemples convergent en $\mathscr{O}(H^2)$ quand H tend vers 0. L'implémentation de ces méthodes est donc validée pour la dimension 2.

Exemples en dimension 3.

Exemple 5.6 *Afin de tester l'implementation des conditions aux limites de Dirichlet, on considère dans cet exemple que :*

$$\begin{aligned} f &= 0,\\ \Gamma_D &= \partial\Omega,\\ g_D &= \sin(\pi x)\sin(\pi y)\sinh(\sqrt{2}\pi z). \end{aligned}$$

La solution du problème est alors :

$$C = \sin(\pi x)\sin(\pi y)\sinh(\sqrt{2}\pi z).$$

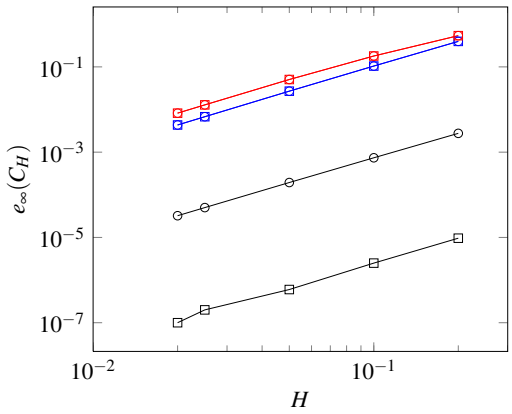


FIGURE 5.12 – Erreur $e_\infty(C_H)$ en fonction de H pour les Exemples 5.3 (—○—), 5.4 (—○—) et 5.5 (—○—) pour les méthodes d'Éléments Finis (—○—) et de Galerkin discontinue (—□—).

Exemple 5.7 *On note $\Gamma_{z,m}$ et $\Gamma_{z,p}$ les faces du domaine Ω pour lesquelles on a $\{z=0\}$ et $\{z=1\}$ respectivement. Afin de tester l'implémentation des conditions aux limites de Neumann, on considère dans cet exemple que :*

$$\begin{aligned} f &= 0, \\ \Gamma_N &= \Gamma_{z,m} \sqcup \Gamma_{z,p}, \\ g_N &= \frac{\partial}{\partial \mathbf{n}} \left(\sin(\pi x)\sin(\pi y)\sinh(\sqrt{2}\pi z) \right), \\ \Gamma_D &= \partial\Omega \setminus \Gamma_N, \\ g_D &= \sin(\pi x)\sin(\pi y)\sinh(\sqrt{2}\pi z). \end{aligned}$$

La solution du problème est alors :

$$C = \sin(\pi x)\sin(\pi y)\sinh(\sqrt{2}\pi z).$$

Exemple 5.8 *Afin de tester l'implémentation du terme source, on considère dans cet exemple que :*

$$\begin{aligned} f &= 3\pi^2 \sin(\pi x)\sin(\pi y)\sin(\pi z), \\ \Gamma_D &= \partial\Omega, \\ g_D &= 0. \end{aligned}$$

La solution du problème est alors :

$$C = \sin(\pi x)\sin(\pi y)\sin(\pi z).$$

On utilise pour critère de validation le taux d'erreur définit en (5.7).

La Figure 5.13 présente l'évolution de l'erreur $e_\infty(C_H)$ en fonction de H lors de la résolution par la méthode d'Éléments Finis (cercle) et de Galerkin discontinue (carré) des des exemples précédents : Exemple 5.6 (—○— et —□—), Exemple 5.7 (—○— et —□—) et Exemple 5.8 (—○— et —□—).

Tous les exemples convergent en $\mathscr{O}(H^2)$ quand H tend vers 0, que ce soit pour la méthode d'Éléments Finis ou celle de Galerkin discontinue. L'implémentation de ces méthodes est donc validée pour la dimension 3.

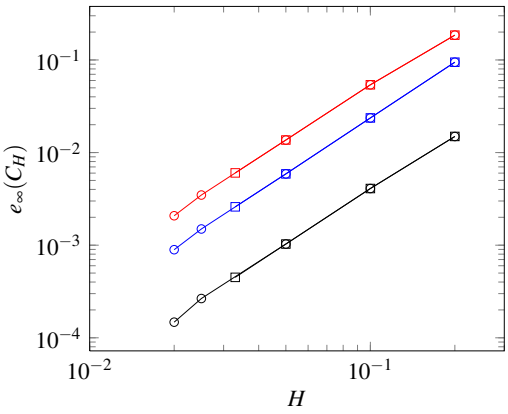


FIGURE 5.13 – Erreur $e_\infty(C_H)$ en fonction de H pour les Exemples 5.6 (—○—), 5.7 (—○—) et 5.8 (—○—) pour les méthodes d'Éléments Finis (—○—) et de Galerkin discontinue (—□—).

Chapitre 6

Mise en œuvre dans le cas $3D$: générations et opérations sur les maillages.

Introduction.

En quittant le cadre de la maquette Python indépendante pour intégrer les méthodes multi-échelles Q_1/*VFDiam* et GD/*VFDiam* au sein du code de calcul *MPCube*, on a laissé de côté *de facto* l'utilisation de maillages quadrangulaires réguliers à l'échelle fine. Compte tenu de leurs propriétés, ces maillages n'avaient pas besoin d'une structure informatique complexe. En effet, une fois un ordre de numérotation des noeuds et des volumes choisi, les connectivités entre éléments (volumes à noeuds et volumes à volumes) sont explicitement connues. Nul besoin donc de générer ou de stocker ces informations. De même, le support des champs de diffusivité et des solutions des problèmes de cellules est connu. Il suffit donc de stocker les seules valeurs des champs pour pouvoir les reconstruire à volonté.

Ce n'est plus le cas au sein des simulations *MPCube* : les maillages y sont quelconques et peuvent contenir plusieurs types de discrétisations (tétraèdres, héxaèdres). *MPCube* ne s'occupe en aucun cas de la construction de ces maillages, son rôle se concentrant sur la résolution des problèmes de diffusion. Pour chaque domaine de simulation Λ, on doit donc fournir un maillage fin $T_h(\Lambda)$ au logiciel *MPCube*.

Dans tout ce chapitre, on reprend les notations du chapitre §3, notamment les Définitions 3.1, 3.2 et 3.4. Le domaine de travail Ω suit donc un découpage $\mathscr{D}$ qui forme le maillage grossier $T_H(\Omega)$. On appelle macroéléments les éléments cubiques de $T_H(\Omega)$. À partir d'un macroélément K, on construit la cellule $\hat{K}$ associée en ajoutant à K une fraction ρ de ses voisins. La Figure 3.1 présente l'exemple d'un macroélement et de sa cellule. Pour chaque macroélement de $T_H(\Omega)$, on doit donc

discrétiser la cellule $\hat{K}$ pour obtenir un maillage $T_h(\hat{K})$.

Ce chapitre présente l'ensemble des opérations et solutions techniques mises en œuvre afin de construire les maillages de milieu nécessaires à *MPCube*. La mise en œuvre des méthodes multi-échelles Q_1/*VFDiam* et GD/*VFDiam* au sein du code *MPCube* est l'objet du chapitre précédent §5.

On détaille tout d'abord les différents outils et fonctionnalités nécessaires à la bonne génération des maillages. On présente ensuite la plate-forme *SALOME* [163], l'outil utilisé pour construire et manipuler les maillages dans ces travaux de thèse.

Les sections suivantes traitent de la construction automatique des maillages $T_h(\hat{K})$ des cellules. La section §6.2 se focalise sur la construction rapide et la discrétisation d'une cellule $\hat{K}$. On décrit à la section §6.3 un cas plus complexe où l'on souhaite que les maillages de cellules voisines coïncident sur les frontières des macroéléments correspondant. Ces maillages coïncidents sont en effet nécessaire à l'utilisation, à l'échelle grossière, des méthodes de Galerkin discontinues, décrites à la section §3.3.2.

Enfin, on présente à la section §6.4 un algorithme permettant de construire plusieurs maillages dérivés les uns des autres, ceci afin de modéliser facilement les phénomènes de dégradation d'un matériau cimentaire.

6.1 La plate-forme *SALOME*.

6.1.1 Description des besoins d'une chaîne de calcul multi-échelle.

6.1.1.1 Maillages.

La vocation première de ces travaux de thèse est de concevoir et d'implémenter des méthodes multi-échelles dédiées aux matériaux cimentaires. Toute implémentation de ces méthodes est donc tenue de pouvoir mailler différentes modélisations de ces matériaux. Comme on l'a décrit à la section §3.6, on représente ici les matériaux cimentaires par une distribution complexe d'objets géométriques, que l'on nomme *inclusions* dans un milieu homogène, *la matrice*. À la section §4.3.2, on a présenté deux méthodes pour représenter ces matériaux dans le cadre de travail de la maquette Python : un milieu plan, des inclusions sphériques et des maillages quadrangulaires réguliers. On travaille ici dans un cadre plus général qui apporte des nouvelles contraintes, le milieu est de dimension 2 ou 3 ; les inclusions ont des formes quelconques ; mais aussi de nouvelles libertés : les maillages fins ne sont pas réguliers.

La première méthode de modélisation, par *dichotomie*, n'est pas plus efficace dans ce contexte élargi. En effet, la forme de la discrétisation ne règle pas le problème majeur de cette méthode puisqu'il n'y aura toujours qu'un très faible nombre de volumes strictement à l'intérieur des petites inclusions. Le milieu sera donc autant déformé que dans le cadre $2D$ régulier. La seconde méthode, par *moyenne* pour-

rait théoriquement être utilisée ici. Elle demanderait cependant un important travail sur les intersections entre les géométries des inclusions, de forme quelconque, et des éléments du maillage (quadrangles ou tétraèdres). Sa mise en application semble donc difficilement réalisable.

On a donc décidé de changer de philosophie et de travailler sur une approche de type Conception Assistée par Ordinateur (ou CAO), couramment utilisée pour travailler sur des géométries complexes aussi bien dans le milieu de la recherche (*Code_Aster* [77], *Cast3M* [76], etc.) que dans le monde industriel (*Abaqus FEA* [6], *Nastran* [146]). Il s'agit de constuire une représentation géométrique du milieu, puis d'en calculer un maillage *adapté*. Tout élément du maillage, face ou volume, appartient alors à un unique objet physique : la matrice, une inclusion, un bord du domaine. La disposition générale des mailles respecte les frontières entre les objets, aux erreurs de discrétisation près.

6.1.1.2 Automatisation.

La méthode multi-échelle suppose de découper le domaine de travail en cellules. Pour un domaine donné, ce n'est donc pas un maillage qu'il faut générer, mais autant de maillages que de cellules, qui sont généralement très nombreuses. On rappelle, pour l'exemple, qu'une simple division d'un domaine $3D$ en dix parties selon chaque direction de l'espace conduit à la création d'un millier de cellules, pour un maillage grossier de seulement 11^3 degrés de liberté. Il devient donc vital de pouvoir automatiser la création des maillages, sous peine de ne pouvoir appliquer les méthodes multi-échelles qu'à de très petits découpages.

6.1.1.3 Parallélisme.

Au sein d'un milieu, les cellules sont physiquement liées les unes aux autres par les frontières des macroéléments sous-jacents. Le reste de la cellule est indépendant de ses voisins et il en est de même pour les maillages associés. Une fois discrétisées les frontières des macroéléments, mailler les cellules peut donc être effectué en parallèle, tout un jeu de maillages de cellules étant réalisé en même temps. Pour profiter de cette possibilité, il est donc important que les outils utilisés soit capable de travailler dans un cadre parallèle.

6.1.1.4 Post-traitement.

Une fois les maillages générés, d'autres problèmes restent à résoudre. En effet, la majeure partie des traitements qui s'avéraient triviaux sur des maillages réguliers deviennent plus complexes à mettre en œuvre dans un cadre quelconque. C'est le cas notamment du recollement des maillages qui va demander une renumérotation complète des maillages à fusionner. De même l'accès aux solutions et aux erreurs

ne se fait plus automatiquement : une coupe de la solution demandera ainsi de couper géométriquement le maillage par le plan de coupe, puis de projeter les valeurs de la solution sur ce plan. Sans être rédhibitoires, ces problèmes demandent l'utilisation d'outils particuliers qu'il faudra donc mettre en place.

6.1.2 Présentation de la plate-forme *SALOME*.

On a choisi de générer les maillages par le biais de la plate-forme *SALOME* [163], développée par le consortium Open CASCADE en collaboration avec le CEA et EDF. Il s'agit d'un logiciel *libre*, distribué sous licence publique générale limitée GNU (ou *GNU LGPL* [109]), qui fournit une plate-forme générique pour les travaux en amont et en aval des simulations numériques.

En amont, *SALOME* peut être utilisée en tant que logiciel de CAO afin de construire des géométries complexes, ou encore modifier des géométries importées d'autres logiciels. Ces géométries peuvent ensuite être discrétisées en utilisant des mailleurs libres (NETGEN [147]) ou commerciaux (GHS3D [103], GHS3D parallel, BLSURF [131], Hexotic [116]).

En aval des simulations, *SALOME* permet de réaliser de nombreuses opérations sur les maillages (fusion, découpage), et sur les champs solutions (visualisations, coupes).

SALOME a été développée dans l'optique d'être pilotable *via* des commandes Python [155]. La version graphique de l'interface utilisateur (GUI Graphic User Interface) ne permet d'ailleurs pas d'accéder à toutes les fonctionnalités, et il faut donc utiliser l'interface texte utilisateur Python (TUI Text User Interface) pour profiter pleinement de certains outils.

Il est à noter que la plate-forme *SALOME* contient le module YACS. Il s'agit d'un module d'interfaçage permettant de piloter des programmes externes à *SALOME*, ou dépendant d'autres modules. En particulier, il permet de paralléliser la construction des maillages de cellules. Cette fonctionnalité n'a cependant pas été utilisée dans le cadre de ces travaux de thèse, faute de temps pour la maitriser.

Remarque 6.1 *Dans le cadre de cette thèse, deux versions de* SALOME *ont été utilisées : la version 4.1.3 sous Mandriva 2006, puis la version 5.1.4 sous Mandriva 2008. Le développement, la correction et l'amélioration de* SALOME *étant très actifs, il se peut que certaines des solutions décrites dans la suite de ce chapitre ne soient plus correctes, ou soient inutilement complexes. Elles méritent néanmoins d'être présentées, car leur valeur pédagogique reste intacte. En effet, il s'agit de solutions originales à des problèmes survenant régulièrement en CAO : automatisation (§6.2), optimisation de routines (§6.2.3), gestion de cas critiques (§6.2.4), etc. Elles font en outre appel à de très nombreuses routines natives de* SALOME*, et représentent donc une source précieuse d'informations pour toute personne désirant utiliser intensivement* SALOME*. À charge au lecteur motivé d'adapter les*

solutions présentées à la dernière version du logiciel, en s'appuyant par exemple sur leurs codes sources, reproduits à l'Annexe §B.

6.1.3 Composants utilisés en amont des simulations numériques.

On présente maintenant les principes généraux qui régissent le fonctionnement de *SALOME* en amont des simulation numériques, c'est-à-dire l'ensemble des opérations qui permettent de passer d'une information sur le milieu (géométrie, diffusivité) à un maillage exploitable par le solveur *MPCube*.

Ce travail s'appuie sur deux modules, les modules GEOM et MESH, qui gèrent respectivement la géométrie du milieu et les maillages. Chacun va permettre de créer une représentation du domaine, respectivement une géométrie et un maillage, qui seront intrinsèquement liés.

La première étape de tout travail est de créer la représentation géométrique du milieu. *SALOME* permet de créer facilement toutes sortes d'objets, soit grâce à l'interface graphique (GUI), soit *via* des commandes Python. Cette dernière solution étant d'ailleurs la seule utilisable si on souhaite travailler sur des formes complexes ou automatiser certaines tâches, elle est à privilégier autant que possible. L'objectif de cette étape est d'obtenir, au fil des manipulations, un *unique* objet représentant le domaine, dans lequel chaque zone caractéristique (frontières, faces de bords, différentes zones de diffusivités) est représentée par un groupe géométrique. Du point de vue CAO, chaque groupe peut contenir un ou plusieurs objets, mais il ne représente qu'une seule réalité physique.

Une fois cet objet créé, on peut le discrétiser *via* le module MESH qui construit un objet maillage lié à l'objet géométrique. Il suffit ensuite de choisir et lancer les algorithmes mailleurs (NETGEN, etc.) pour obtenir un maillage. À partir des groupes géométriques, on peut alors former des groupes d'éléments : groupes de volumes pour les solides, groupes de faces pour les bords. Ceux-ci sont indispensables à tout travail de simulation sur le maillage, puisqu'ils servent notamment à marquer les éléments du maillages concernés par les conditions aux limites.

Il est important de noter que l'objet géométrique est indispensable au bon fonctionnement de l'objet maillage. Si on s'avisait de supprimer la géométrie associée, toute opération au-delà de l'affichage et du stockage du maillage courant deviendrait impossible. Notamment, on ne pourrait ni recalculer le maillage, ni former des groupes ni supprimer des éléments du maillage autrement qu'à la main. Sur ce dernier point, on pourra se reporter à la section §6.4 pour avoir plus de détails.

Par expérience personnelle, l'étape de création géométrique est de loin la plus importante des deux. En effet, même si créer le maillage fait intervenir des algorithmes complexes, gourmands en mémoire et en temps, leur mise en œuvre est aisée au sein de la plate-forme *SALOME*. La création géométrique, elle, demande une réflexion beaucoup plus importante, notamment pour la création des groupes géométriques si utiles pour la division du maillage en familles. De plus, certaines

routines natives de *SALOME* ne tiennent pas compte des besoins spécifiques liés à la modélisation des matériaux cimentaires et demandent à être adaptées. Plus loin dans ce chapitre, on présente en détail la construction de la géométrie d'une cellule où les inclusions sont groupées par type (§6.2), ainsi qu'un exemple concret où l'on a dû adapter une routine *SALOME* (§6.2.3).

6.1.4 Composants utilisés en aval des simulations numériques.

La plate-forme *SALOME* dispose de plusieurs outils utilisés pour post-traiter les simulations numériques *MPCube*. L'outil MEDSPLITTER, par exemple, permet de manipuler des fichiers au format MED. Comme son nom le laisse supposer, sa fonction première est de diviser les maillages en plusieurs morceaux, par exemple pour effectuer des simulations en parallèle.

MEDSPLITTER est utilisé à deux étapes de la chaîne de calcul multi-échelle *SALOME-MPCube*. Tout d'abord, il permet d'extraire le maillage $T_h(K)$ d'un macro-élément K du maillage $T_h(\hat{K})$ de la cellule associée. Les champs solutions associés sont extraits en même temps.

Ensuite, une version adaptée de MEDSPLITTER permet de *fusionner* plusieurs maillages ensemble, et non plus de les diviser. Cette nouvelle fonctionnalité permet d'obtenir un unique fichier contenant la solution d'un problème multi-échelle sur un patch de macroéléments.

SALOME contient également le module POST-PRO qui permet entre autres de manipuler les champs solutions contenus dans un fichier MED. Il est donc utilisé pour afficher les solutions, tracer des coupes et calculer des taux d'erreurs.

6.2 Génération automatique des maillages de cellules.

6.2.1 Notations.

Soit d la dimension de l'espace $\mathbb{R}^d$ de travail, où $d = 2$ ou $d = 3$. Dans tout ce qui suit, on utilise un vocabulaire et des notations liés à la dimension 3 : on parle de *volumes* et de *faces* pour désigner les éléments de dimension d et $d-1$ d'un maillage, les coordonnées sont des triplets, etc. Les notations sont cependant facilement adaptables au cas de la dimension 2.

Tout au long de ce chapitre, on travaille sur une domaine Ω parallélépipédique, l'ouvert $]0,H(1)[\times]0,H(2)[\times]0,H(3)[$ où H est un triplet de réels strictement positifs. On reprend les notations introduites à la section §3.1, notamment le *découpage* $(\mathscr{D},\rho)$ de Ω (Def. 3.1) en *macroéléments* K (Def. 3.2), construisant ainsi le maillage grossier $T_H(\Omega)$. On utilise également la construction et la numérotation des *cellules* $\hat{K}$ (Def 3.4) associées aux éléments K de $T_H(\Omega)$.

À cet aspect topologique, on va maintenant ajouter des définitions plus géométriques, nécessaires à la construction des objets géométriques de *SALOME*.

Définition 6.1 *Soit O un objet borné. On définit le* point inférieur m_O *et le* point supérieur M_O *de O par les relations suivantes :*

$$\forall 1 \leq i \leq d \qquad m_O(i) = \inf\{x(i), x \in O\} \tag{6.1}$$

$$\forall 1 \leq i \leq d \qquad M_O(i) = \sup\{x(i), x \in O\} \tag{6.2}$$

Définition 6.2 *On appelle* inclusion*, et on note I, un ouvert borné régulier de* $\mathbb{R}^3$ *situé tout ou partie dans* Ω*. On note* ∂I *sa frontière telle qu'on la définit en topologie générale [51]. On note* $\complement I$ *le complémentaire de* $I \sqcup \partial I$ *dans* $\mathbb{R}^3$*. I,* ∂I *et* $\complement I$ *vérifient donc :*

$$I \sqcup \partial I \sqcup \complement I = \mathbb{R}^3. \tag{6.3}$$

En pratique, les inclusions modélisent une réalité physique, concrête, et sont donc des solides : polyhèdres, sphères, etc. On suppose donc les inclusions suffisament régulières pour que des opérations de coupe et d'union restent dans la géométrie usuelle. En particulier, on suppose que la frontière ∂I, qui représente la surface de l'inclusion, est une variété compacte sans bord [98] et que les complémentaires vérifient la propriété suivante :

$$\forall A, B \qquad \complement(A \cap B) = \complement A \cup \complement B. \tag{6.4}$$

Définition 6.3 *On note* $\mathscr{I}$ *l'ensemble des inclusions du milieu, et on les supposera disjointes les unes des autres. Deux inclusions peuvent cependant être tangentes, c'est-à-dire partager tout ou partie de leurs frontières.*

Définition 6.4 *On appelle* macroélément homogène*; et on note* K_0*; un parallélépipède rectangle issu du découpage grossier de notre domaine de travail* Ω*. Ses segments suivent les axes des coordonnées et on pose* $m_{K_0} = (x_m, y_m, z_m)$ *et* $M_{K_0} = (x_M, y_M, z_M)$*. Il s'agit en fait du macroélément K privé de ses inclusions.*

Définition 6.5 *On appelle* cellule homogène*; et on note* $\hat{K}_0$*; l'homothétie du macroélément homogène, de rapport* $1 + 2\rho$ *et de centre* $C = \frac{1}{2}(m_{K_0} + M_{K_0})$ *de* K_0*. Il s'agit donc de la cellule* $\hat{K}$ *dont on a ôté les inclusions.*

Définition 6.6 *On appelle* couronne homogène*, et on note* K_0^c*; le solide s'étendant entre les faces du macroélément homogène* K_0 *et de la cellule homogène* $\hat{K}_0$*. Il occupe donc la zone de recouvrement de la cellule* $\hat{K}$*.*

Définition 6.7 *On note* $\mathscr{I}^K$ *l'ensemble des inclusions d'intersection non vide avec la cellule* $\hat{K}$*.*

6.2.2 Principes généraux.

On suppose disposer d'une description du milieu précisant la forme et la position de l'ensemble $\mathscr{I}$ des inclusions s'y trouvant. Cette description peut reproduire un échantillon physique d'un matériau cimentaire, à partir d'une image obtenue par microscopie optique [83], microscopie électronique à balayage [165] ou microtomographie [101, 129, 136].

Elle peut également être générée par le logiciel COMBS que l'on a décrit à la section §4.3.1.

On choisit alors un découpage $(\mathscr{D}, \rho)$ du domaine. Celui-ci fixe le nombre de macroéléments selon chaque direction physique, construisant ainsi le maillage grossier $T_H(\Omega)$. Le taux de sur-échantillonnage ρ permet de construire la cellule $\hat{K}$ pour chaque macroélement $K \in T_H(\Omega)$.

L'Algorithme 6.1 présente la manière dont on obtient les maillages $T_h(\hat{K})$ des cellules à partir de ces seules informations, les différentes étapes étant décrites par la suite. Cet algorithme est illustré à la Figure 6.1. On laisse pour l'instant de côté les problèmes de raccordement le long des frontières des macroéléments, qui font l'objet d'une étude détaillée à la section §6.3.

Algorithme 6.1 Génération des maillages de cellules $T_h(\hat{K})$.

1: Répartir les inclusions entre les cellules. ▷ cf. §6.2.2.1
2: **Pour** chaque K de $T_H(\Omega)$ **faire**
3: Créer $\left\{K_0, K_0^c\right\}$, les zones homogènes de la cellule. ▷ cf. §6.2.2.2
4: Créer les inclusions $I \in \mathscr{I}^K$ de la cellule.
5: Assembler la cellule géométrique $\hat{K}$. ▷ cf. §6.2.2.3
6: Marquer les groupes d'inclusions au sein de la cellule. ▷ cf. §6.2.2.4
7: Marquer les groupes de bords au sein de la cellule.
8: Mailler la cellule $\hat{K}$. ▷ cf. §6.2.2.5
9: Marquer les familles d'éléments à partir des groupes correspondants.
10: Exporter le maillage au format MED.
11: **Fin Pour**

6.2.2.1 Répartition des inclusions.

La première étape de l'Algorithme 6.1 consiste à déterminer quelles inclusions se trouvent dans, ou à proximité, de chaque cellule. L'important est de réduire la liste complète $\mathscr{I}$ des inclusions du domaine, qui peut-être très grande, au nombre beaucoup plus faible des inclusions $\mathscr{I}^K$ liées à une cellule donnée. Cette sélection n'a pas besoin d'être parfaite, mais seulement de reconnaître correctement les inclusions de chaque cellule, c'est-à-dire de vérifier la propriété suivante :

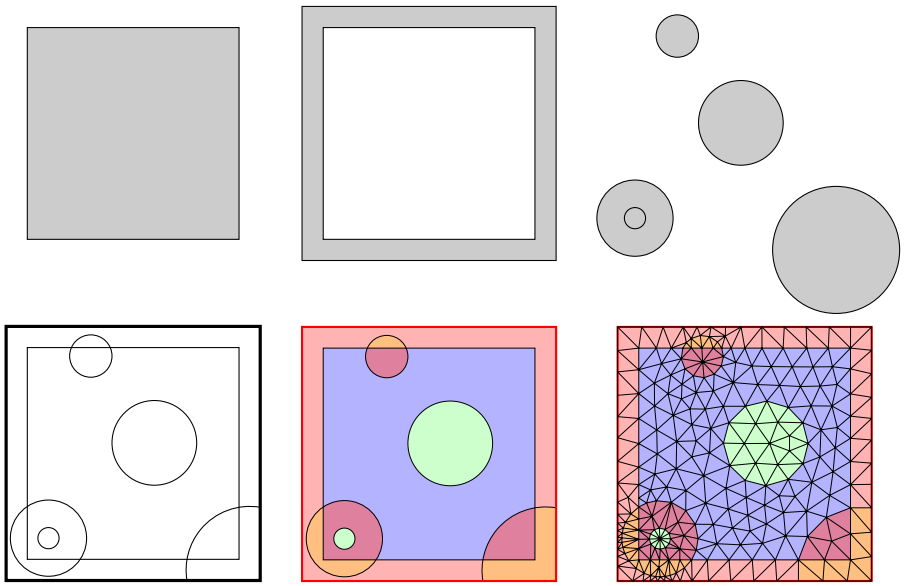


FIGURE 6.1 – Illustration du travail de l'Algorithme 6.1 sur une cellule. En haut, de gauche à droite, les objets indépendants : le macroélément homogène, la couronne homogène et les inclusions. En bas, de gauche à droite, la cellule assemblée en un unique objet, le marquage des groupes d'objets (chaque couleur représente un groupe, les faces de bords étant en rouge) et le maillage de la cellule.

$$\forall I \in \mathscr{I}, \forall \hat{K} \qquad I \cap \hat{K} \neq \emptyset \Rightarrow I \in \mathscr{I}^K \tag{6.5}$$

On peut donc se permettre d'intégrer *en plus* certains éléments à la liste des inclusions de la cellule. Ces inclusions surnuméraires sont naturellement écartées lors de la construction de la cellule. Elles ne nuisent donc pas au bon fonctionnement de l'algorithme.

L'Algorithme 6.2 présente la construction, pour chaque cellule $\hat{K}$ du découpage, de l'ensemble des inclusions $\mathscr{I}^K$. On rappelle que Ω est un domaine parallélépipédique $\prod_{i=1}^{d}]0, H(i)[$ où H est un triplet de réels strictement positifs. $\lfloor\rfloor$ et $\lceil\rceil$ sont les symboles usuels respectifs de la partie entière par défaut et de la partie entière par excès.

La Définition 3.2 attribue à tout macroélément K des coordonnées dans $\mathbb{N}^d$, que l'on note également K. On utilise ici la relation d'ordre partiel $\leq$ sur les macroéléments ; et donc sur les cellules associées ; définie par la Définition 3.3.

Algorithme 6.2 Répartition des inclusions $\mathscr{I}$ par cellule.

Entrée: L'ensemble $\mathscr{I}$ des inclusions du domaine.
Sortie: Les ensembles $(\mathscr{I}^K)_{K \in T_H(\Omega)}$.

1: Pour tout macroélément K, on pose $\mathscr{I}^K = \emptyset$.
2: **Pour** chaque inclusion $I \in \mathscr{I}$ **faire**
3: Calculer m_I et M_I. ▷ cf. Déf. 6.1
4: **Pour** tout $1 \leq i \leq d$ **faire**
5: $\hat{K}_{m,I}(i) = \left\lceil \frac{m_I(i)}{H(i)} - \rho - 1 \right\rceil$.
6: $\hat{K}_{M,I}(i) = \left\lfloor \frac{M_I(i)}{H(i)} + \rho \right\rfloor$.
7: **Fin Pour**
8: **Pour** toute cellule $\hat{K}_{m,I} \leq \hat{K} \leq \hat{K}_{M,I}$ **faire**
9: Ajouter I à $\mathscr{I}^K$.
10: **Fin Pour**
11: **Fin Pour**

Preuve de la correction de l'Algorithme 6.2:

Soit $I \in \mathscr{I}$, et $\hat{K}$ une cellule, tel que $I \cap \hat{K} \neq \emptyset$. Soit $x \in I \cap \hat{K}$, x vérifie :

$$\forall 1 \leq i \leq d \qquad m_I(i) \leq x(i) \leq M_{\hat{K}}(i).$$

Or on connaît explicitement la géométrie de la cellule $\hat{K}$ en fonction de ses coordonnées entières dans le maillage grossier $T_H(\Omega)$. Il vient donc :

$$\forall 1 \leq i \leq d \qquad M_{\hat{K}}(i) = (\hat{K}(i) + 1 + \rho)H(i).$$

On a donc :

$$\forall 1 \leq i \leq d \qquad \frac{m_I(i)}{H(i)} - 1 - \rho \leq \hat{K}(i).$$

Les coordonnées de $\hat{K}$ étant à valeur entière, il vient par passage à la partie entière par excès :

$$\forall 1 \leq i \leq d \qquad \hat{K}_{m,I}(i) \leq \hat{K}(i).$$

Et donc, par définition de l'ordre partiel $\leq$ (cf. Définition 3.3) sur les macro-éléments et les cellules, on a :

$$\hat{K}_{m,I} \leq \hat{K}.$$

De même, on montre que :

$$\hat{K} \leq \hat{K}_{M,I}.$$

L'Algorithme 6.2 place donc I dans l'ensemble $\mathscr{I}^K$, ce qui prouve la correction de l'algorithme.

■

6.2.2.2 Construction des objets élémentaires.

On peut maintenant travailler sur chaque cellule indépendamment des autres. Le travail suit les grandes lignes de la philosophie *SALOME* telle qu'énoncée à la section §6.1.3.

On crée tout d'abord les objets géométriques correspondant aux différents éléments de la cellule. En reprenant les notations de la section §6.2.1, il y a tout d'abord les zones homogènes de la cellule, c'est-à-dire K_0, le cube délimitant le macroélément, et K_0^c la couronne délimitant la zone de sur-échantillonnage. Il y a ensuite l'ensemble $\mathscr{I}^K$ des inclusions de la cellule. Les inclusions de formes simples (sphères, cubes, etc.) sont créées par des routines natives de *SALOME*, tandis que les plus complexes sont l'œuvre de routines spécifiques.

6.2.2.3 Assemblage de la cellule.

On assemble ensuite la cellule *via* une commande *SALOME* native portant le nom de MAKEPARTITION. Celle-ci a pour rôle de découper les zones homogènes de la cellule, les objets *sources* selon la terminologie *SALOME*, en fonction des inclusions, que *SALOME* nomme objets *outils*.

On obtient alors une unique géométrie, celle de la cellule $\hat{K}$, ayant intégrée les informations des deux ensembles d'objets, c'est-à-dire possédant les surfaces et arêtes des zones homogènes et des inclusions $\mathscr{I}^K$, comme on peut le voir sur la Figure 6.1 (en bas, à gauche).

6.2.2.4 Marquage des objets.

On marque ensuite les formes élémentaires composant la cellule pour former des groupes géométriques. Le classement des inclusions en fonction de leurs propriétés physiques (tailles, diffusivités) est ainsi appliqué aux solides correspondants. De même les surfaces composant les bords de la cellule sont regroupées.

6.2.2.5 Génération du maillage.

La dernière étape du travail consiste à discrétiser l'objet cellule pour obtenir le maillage $T_h(\hat{K})$. On appelle pour cela un des mailleurs interfacés à *SALOME*, par exemple NETGEN. Si l'objet cellule a été correctement construit, le mailleur en construira un maillage adapté. Cependant, il est à noter qu'en fonction de la complexité de la cellule, les ressources nécessaires en temps et en mémoire peuvent être extrêmement importantes, sans garantie sur la qualité du maillage obtenu.

Une fois le maillage $T_h(\hat{K})$ obtenu, on marque les différentes groupes d'éléments à partir des groupes géométriques correspondants : groupes de volumes pour les solides, groupes de faces pour les bords.

SALOME peut exporter les maillages qu'il créé sous plusieurs formats : MED [142], UNV (I-DEAS 10) [149] et DAT (Nastran) [146]. On a retenu pour ces travaux de thèse le format d'échange MED car, d'une part, il est facilement lu par *MPCube*, et d'autre part parce qu'il permet de stocker dans un unique fichier un maillage et les champs discrétisés associés.

On exporte donc le maillage $T_h(\hat{K})$ au format MED. Le fichier ainsi obtenu contient alors toutes les informations nécessaires à une simulation *MPCube* : la discrétisation (éléments et connectivités), les groupes d'éléments (exportées en tant que *familles*) et les bords de la cellule.

6.2.2.6 Validation de Algorithme 6.1.

Afin de valider l'Algorithme 6.1, on génère les maillages fins correspondant à l'exemple suivant :

Exemple 6.1 *On se place dans* $\Omega = [0,1]^3$. *Pour* $(i,j,k) \in \mathbb{N}^3$ *, on définit l'inclusion* $I_{i,j,k}$ *comme la sphère de centre* $C_{i,j,k}$ *et de rayon* $r = 0.03$*, où l'on a posé :*

$$C_{i,j,k} = \begin{bmatrix} i*0.01 \\ j*0.01 \\ z*0.01 \end{bmatrix} \tag{6.6}$$

Pour tout entier n, on définit le domaine Ω_n *comme le milieu* Ω *ajouté de l'ensemble des inclusions* $\mathscr{I}_n$ *:*

$$\mathscr{I}_n = \{I_{i,j,k}, 0 \leq i,j,k \leq n\} \tag{6.7}$$

On ne divise pas le milieu Ω_n et on prend un sur-échantillonnage ρ nul. Une unique cellule s'étendra donc sur tout Ω_n, contenant $N = (n+1)^3$ inclusions. On maille le milieu Ω_n avec BLSURF et GHS3D. La Figure 6.3 présente des captures d'écran de la plate-forme *SALOME* des étapes de travail de l'Algorithme 6.1 sur le domaine Ω_2.

La Figure 6.2 présente le temps nécessaire à la génération des maillages $T_h(\Omega_n)$, pour $0 \leq n \leq 9$. Les différentes courbes permettent de jauger la répartition du temps parmi les étapes de construction : sélection des inclusions, création des objets, assemblage de la cellule et marquage des groupes de cellules. Il apparaît clairement que c'est l'assemblage de la cellule qui prend le plus de temps, loin devant les autres étapes de la construction géométrique.

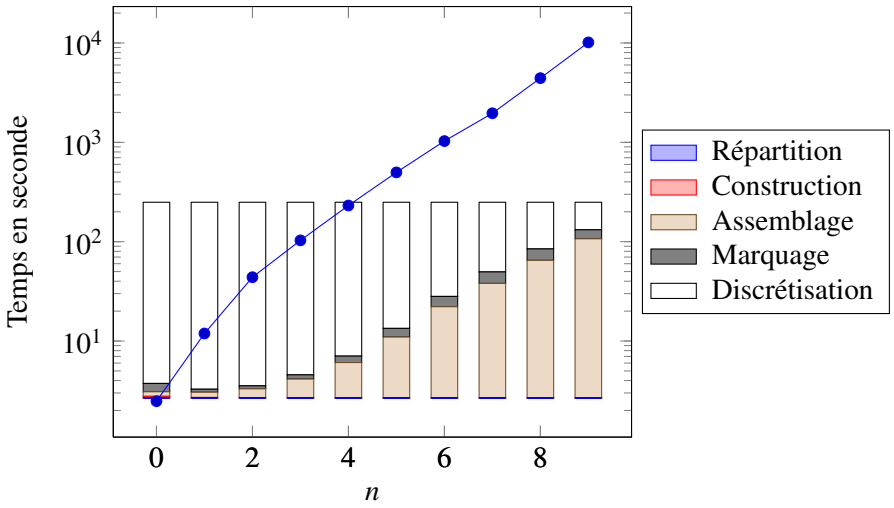


FIGURE 6.2 – Temps de génération, en secondes, des maillages $T_h(\Omega_n)$ en fonction de n. Les barres présentent la répartition du temps total sur les différentes étapes de construction. L'étape d'assemblage apparaît clairement comme étant la plus chronophage.

6.2.3 Adaptation de la fonction native MAKEPARTITION.

6.2.3.1 Description de la fonction native MAKEPARTITION.

La première implémentation de l'Algorithme 6.1 n'utilise que des fonctions natives de *SALOME*, notamment lors de l'assemblage de la géométrie de la cellule (cf. Alg. 6.1 L.5). Or, les différents essais de génération de cellule, en particulier l'Exemple 6.1, montrent que cette étape consomme la plus grande partie du temps CPU consacré à la méthode. Le temps nécessaire augmente en effet de manière exponentielle avec le nombre d'inclusions par cellule.

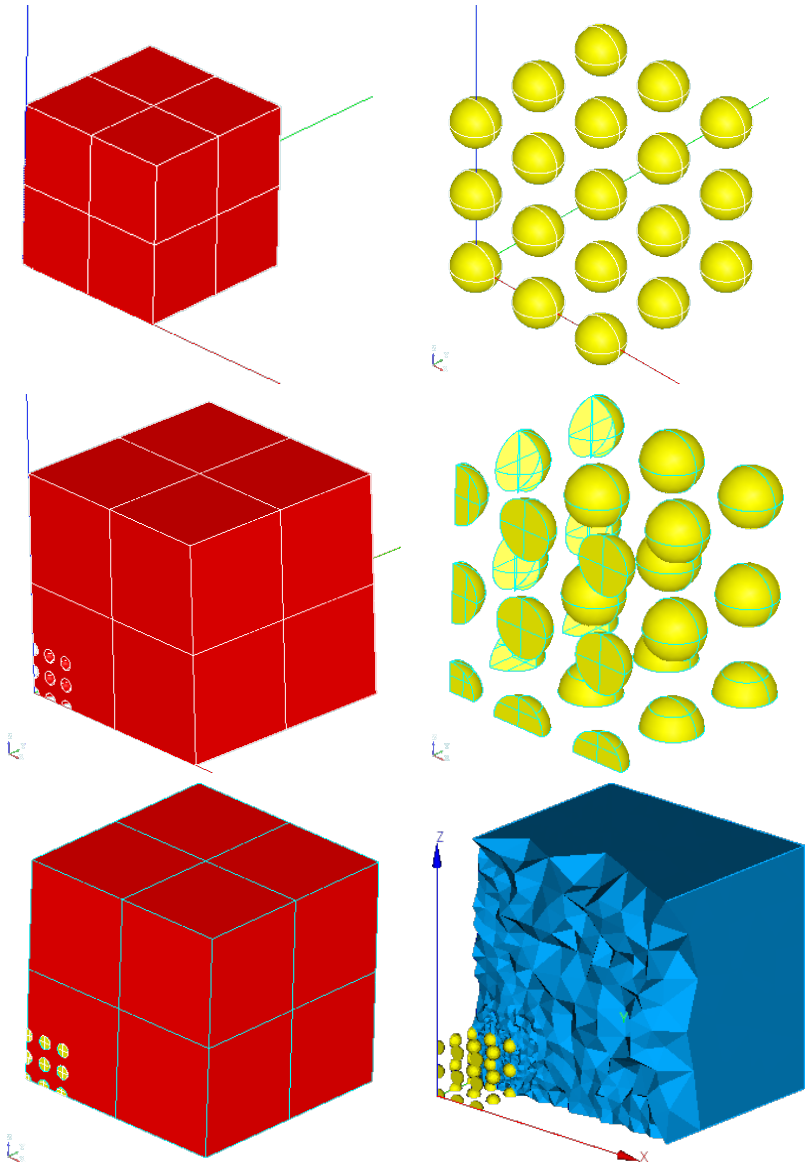


FIGURE 6.3 – Captures d'écran de la plate-forme *SALOME* lors du travail de l'Algorithme 6.1 sur l'Exemple 6.1 pour $n = 2$. L'assemblage de la géométrie de la cellule a été détaillée afin d'illustrer le travail de l'Algorithme 6.3. De gauche à droite et de bas en haut : la zone homogène de macroélément K_0, l'ensemble des inclusions $\mathscr{I}$ (zoom), la zone homogène et les inclusions coupées (zoom), la géométrie de la cellule et enfin le maillage $T_h(\Omega_2)$. Une partie du maillage $T_h(\Omega_2)$ est invisible sur l'image, afin de mettre en valeur la discrétisation des inclusions.

On a donc décidé de réécrire cette étape de la méthode, en évitant cette fois l'appel à la méthode MAKEPARTITION. Cette méthode est très puissante mais effectue un traitement général quelles que soient les inclusions présentes au sein de la cellule. Or la grande majorité de ces inclusions ne requière qu'un traitement très simple. C'est le cas des inclusions *strictement* à l'intérieur du macroélément ou de la zone de sur-échantillonnage, qui n'ont pas d'interactions avec les parois de la cellule.

On a donc décomposé les fonctionnalités de la routine MAKEPARTITION en différentes étapes de travail, présentées à l'Algorithme 6.3. Les ensembles $\mathscr{I}_M^K$, $\mathscr{I}_{MC}^K$, $\mathscr{I}_C^K$ et $\mathscr{I}_{CE}^K$ sont définis à la section §6.2.3.2. En pratique, il est peu probable que le fonctionnement de la routine suive ce découpage. Il ne s'agit là que d'une description des effets apparents de la méthode MAKEPARTITION, c'est-à-dire d'une manière, en partant des mêmes objets, d'arriver au même résultats. Chacune de ces étapes a ensuite été implémentée en tenant compte des spécificités du travail sur les matériaux cimentaires.

Algorithme 6.3 Assemblage de la géométrie d'une cellule.

Entrée: Les zones homogènes $\left\{K_0, K_0^c\right\}$ de la cellule.
Entrée: L'ensemble $\mathscr{I}^K$ des inclusions à placer dans la cellule.
Sortie: La géométrie de la cellule $\hat{K}$.

1: Répartir les inclusions $\mathscr{I}^K$ en fonction de leur position dans la cellule.

2: **Pour** chaque inclusion $I \in \mathscr{I}_{MC}^K$ **faire** ▷ cf. Algorithme 6.4.
3: Calculer $I_{MC} = I \cap K_0$.
4: Calculer $I_{CM} = I \cap K_0^c$.
5: **Fin Pour**
6: **Pour** chaque inclusion $I \in \mathscr{I}_{CE}^K$ **faire** ▷ cf. Algorithme 6.4.
7: Calculer $I_{CE} = I \cap K_0^c$.
8: **Fin Pour**

9: Couper K_0 par les inclusions $\mathscr{I}_M^K$ et $\mathscr{I}_{MC}^K$. ▷ cf. Algorithme 6.5.
10: Couper K_0^c par les inclusions $\mathscr{I}_{MC}^K$, $\mathscr{I}_C^K$ et $\mathscr{I}_{CE}^K$. ▷ cf. Algorithme 6.5.

11: Coller les objets coupés en conservant leurs frontières pour obtenir $\hat{K}$.
12: **Renvoi** $\hat{K}$.

L'implémentation de cet algorithme s'est inspirée des travaux d'Erwan ADAM pour le développement du module COMPS (cf. §4.3.1). Deux idées principales sont appliquées. Tout d'abord, traiter simplement les inclusions strictement à l'intérieur de la cellule, et réserver les traitements complexes à la frange minoritaire des inclusions coupant les parois de la cellule. Ensuite, travailler sur ce que *SALOME* appelle les *shells* des solides, plutôt que directement sur les solides. Il s'agit des

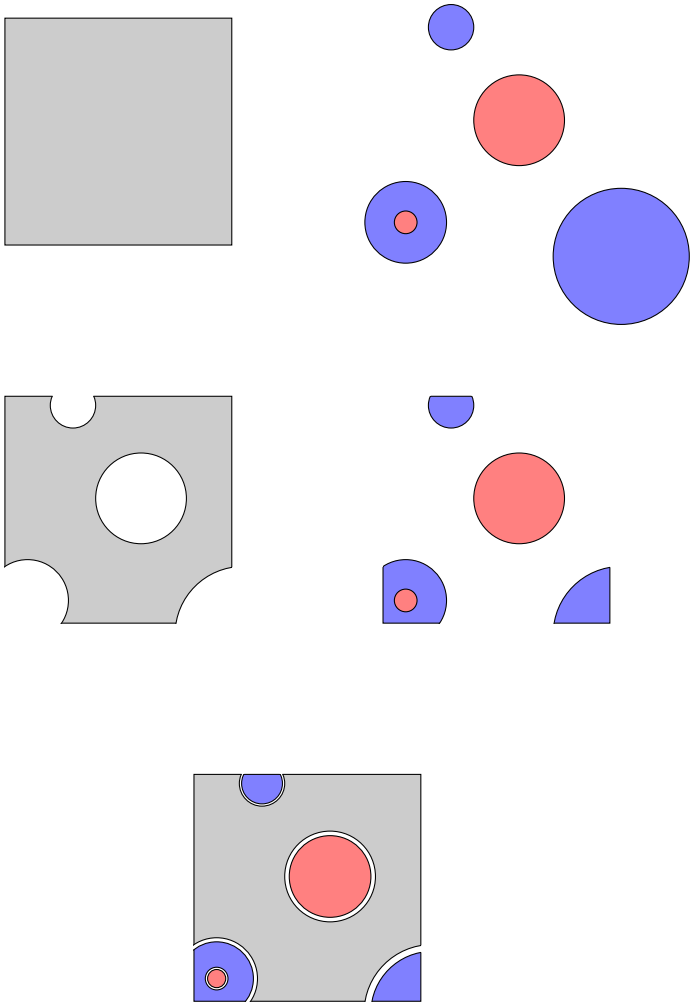


FIGURE 6.4 – Illustration de l'assemblage de la géométrie d'une cellule par l'Algorithme 6.3. Seul le travail sur le macroélément est ici représenté, la zone de recouvrement étant traitée de manière similaire. De gauche à droite et de haut en bas : la zone homogène de macroélément K_0 (en gris), les ensembles des inclusions $\mathscr{I}_M$ (en rouge) et $\mathscr{I}_{MC}$ (en bleu), la zone homogène et les inclusions coupées et la géométrie du macroélement.

surfaces délimitant les solides, d'où leur noms de «*coquilles*» dans la terminologie *SALOME*. Il s'agit en fait de la frontière de l'inclusion, telle qu'on l'a définie à la section §6.2.1.

Travailler sur les frontières plutôt que sur les solides permet d'économiser un temps considérable. En effet, les opérations géométriques comme l'intersection ou la coupe sont moins gourmandes et moins complexes en dimension 2 (opérations frontière à frontière) ou en couplage $2D$-$3D$ (opérations frontière à solide) qu'en $3D$ complet (opérations solide à solide).

Cette façon de faire nécessite deux fonctionnalités supplémentaires : le passage du solide à sa frontière et vice-versa. Le coût d'utilisation de ces deux fonction est réduit en limitant le nombre de leurs appels au strict minimum. Inutile en effet de reconstruire un solide à partir de sa frontière si l'on doit encore effectuer des opérations dessus. C'est dans cette optique qu'on a développé les algorithmes d'intersection (Algorithme 6.4) et de coupe (Algorithme 6.5) présentés plus bas.

Remarque 6.2 *Dans le cas général, la construction d'un solide A à partir de sa frontière ∂A est un problème complexe, qui n'est pas toujours bien posé. En pratique, il est ici le fait d'une fonction* SALOME *native nommée* MAKESOLID. *Dans tout ce qui suit, on supposera que la correction de cette fonction est assurée, sous réserve que la frontière passée en paramètre possède une orientation cohérente. Ce point particulier, bien que passé sous silence dans les algorithmes de ce chapitre, est attentivement vérifié par les implémentations des méthodes correspondantes, disponibles en annexe §B.1.*

6.2.3.2 Répartition des inclusions dans la cellule.

Définition 6.8 *Pour toute inclusion $I \in \mathscr{I}^K$, en utilisant les quantités m_I et M_I définies à la Définition 6.1, on définit la* boîte encadrante $\mathbb{B}(I)$ *par ses sommets* $S = \prod_{1\leq j\leq n}\{m_I(j), M_I(j)\}$.

En fonction de la position de $\mathbb{B}(I)$ vis-à-vis des zones homogènes de la cellule $\hat{K}$, on attribue alors toute inclusion $I \in \mathscr{I}^K$ à l'un des ensembles suivants :

– $\mathscr{I}_M^K$: ensemble des inclusions *strictement* intérieures au macroélément.

$$\mathscr{I}_M^K = \left\{ I \in \mathscr{I}^K, \mathbb{B}(I) \subset K_0 \right\}$$

– $\mathscr{I}_{MC}^K$: ensemble des inclusions à cheval sur le macroélément et la couronne.

$$\mathscr{I}_{MC}^K = \left\{ I \in \mathscr{I}^K, \mathbb{B}(I) \cap K_0 \neq \emptyset, \mathbb{B}(I) \cap K_0^c \neq \emptyset \right\}$$

– $\mathscr{I}_C^K$: ensemble des inclusions *strictement* intérieures à la couronne.

$$\mathscr{I}_C^K = \left\{ I \in \mathscr{I}^K, \mathbb{B}(I) \subset K_0^c \right\}$$

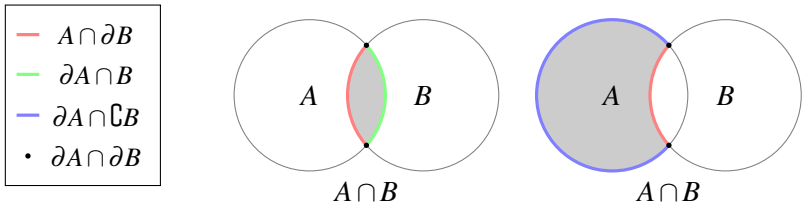


FIGURE 6.5 – Intersection (à gauche) et coupe (à droite) de deux ensembles A et B. On distingue les différentes surfaces caractéristiques utilisées par l'Algorithme 6.4 et l'Algorithme 6.5.

– $\mathscr{I}^K_{CE}$: ensemble des inclusions à cheval sur la cellule et sur l'extérieur de la cellule.

$$\mathscr{I}^K_{CE} = \left\{ I \in \mathscr{I}^K, \mathbb{B}(I) \cap K_0 = \emptyset, \mathbb{B}(I) \cap K_0^c \neq \emptyset, \mathbb{B}(I) \cap \complement K_0^c \neq \emptyset \right\}$$

Remarque 6.3 *Dans certains cas, il peut s'avérer difficile de déterminer à quel ensemble appartient une inclusion. C'est le cas par exemple si une inclusion sphérique est incluse dans le macroélément, mais où sa frontière est tangente à la frontière du macroélément. De plus, à cause des erreurs d'arrondis lors des calculs de placement, le même problème peut se poser si l'inclusion est très proche de la paroi.*

Cette répartition a pour but de séparer les inclusions faciles à traiter de celles pouvant poser problème. Pour cette raison, les cas limites sont systématiquement placés dans un des deux ensembles à traitement complexe $\mathscr{I}^K_{MC}$ ou $\mathscr{I}^K_{CE}$.

6.2.3.3 Intersection d'un grand nombre d'objets avec un unique objet.

L'Algorithme 6.4 détaille la manière dont on obtient rapidement les restrictions à un objet de référence O d'une série de petits objets $(a_i)_{1 \leq i \leq n}$. Au sein de la méthode améliorée de construction de la géométrie de la cellule, il assure les étapes (Alg. 6.3 L.2) et (Alg. 6.3 L.6). Cette méthode a été implémentée en Python dans le cadre de ces travaux de thèse sous le nom MAKEINTERSECTIONBYSHELLS, disponible à l'annexe §B.1.1.

Algorithme 6.4 Intersection d'un grand nombre d'objets avec un unique objet.

Entrée: L'objet de référence O.
Entrée: L'ensemble des objets $(a_i)_{1\leq i\leq n}$ à traiter.
Sortie: L'ensemble des objets $(O\cap a_i)_{1\leq i\leq n}$.

1: Obtenir ∂O.
2: **Pour** chaque objet a_i **faire**
3: Obtenir ∂a.
4: $s_i \leftarrow \partial O\cap a_i$.
5: $t_i \leftarrow O\cap \partial a_i$.
6: $u_i \leftarrow \partial O\cap \partial a_i$.
7: Construire le solide T_i associé à $s_i\cup t_i\cup u_i$.
8: **Fin Pour**
9: **Renvoi** $(T_i)_{1\leq i\leq n}$.

Preuve de la correction de l'Algorithme 6.4:

Pour tous ensembles A et B, en utilisant la propriété de partition (6.3), on a :

$$\partial(A\cap B) = (\partial(A\cap B)\cap A)\sqcup(\partial(A\cap B)\cap\partial A)\sqcup\left(\partial(A\cap B)\cap \complement A\right). \tag{6.8}$$

En utilisant (6.4), il vient :

$$x\in\partial(A\cap B)\cap A \Leftrightarrow \begin{cases} x\notin \complement(A\cap B)\\ x\notin A\cap B\\ x\in A\end{cases} \Leftrightarrow \begin{cases} x\notin \complement A\cup\complement B\\ x\notin B\\ x\in A\end{cases} \Leftrightarrow x\in A\cap\partial B,$$

et

$$x\in\partial(A\cap B)\cap\complement A \Leftrightarrow \begin{cases} x\notin \complement A\cup\complement B\\ x\notin B\\ x\in \complement A\end{cases} \Leftrightarrow x\in\emptyset.$$

En distribuant selon B le second ensemble de (6.8), il vient :

$$\partial(A\cap B)\cap\partial A = (\partial(A\cap B)\cap\partial A\cap B) \sqcup(\partial(A\cap B)\cap\partial A\cap\partial B)\sqcup\left(\partial(A\cap B)\cap\partial A\cap\complement B\right).$$

On a alors :

$$x\in\partial(A\cap B)\cap\partial A\cap B \Leftrightarrow \begin{cases} x\notin \complement A\cup\complement B\\ x\notin A\cap B\\ x\notin A\cup\complement A\\ x\in B\end{cases} \Leftrightarrow \begin{cases} x\notin A\cup\complement A\\ x\in B\end{cases} \Leftrightarrow x\in\partial A\cap B,$$

ainsi que,

$$x \in \partial(A \cap B) \cap \partial A \cap \partial B \Leftrightarrow \begin{cases} x \notin \complement A \cup \complement B \\ x \notin A \cap B \\ x \notin A \cup \complement A \\ x \notin B \cup \complement B \end{cases} \Leftrightarrow \begin{cases} x \notin A \cup \complement A \\ x \notin B \cup \complement B \end{cases} \Leftrightarrow x \in \partial A \cap \partial B,$$

et

$$x \in \partial(A \cap B) \cap \partial A \cap \complement B \Leftrightarrow \begin{cases} x \notin \complement A \cup \complement B \\ x \notin A \cap B \\ x \notin A \cup \complement A \\ x \in \complement B \end{cases} \Leftrightarrow x \in \emptyset.$$

Finalement, il vient :

$$\partial(A \cap B) = (A \cap \partial B) \sqcup (\partial A \cap B) \sqcup (\partial A \cap \partial B). \tag{6.9}$$

Appliquée aux objets O et a_i, l'équation précédente permet d'assurer :

$$\partial(O \cap a_i) = s_i \cup t_i \cup u_i = \partial T_i, \tag{6.10}$$

ce qui assure la correction de l'Algorithme 6.4.

■

6.2.3.4 Coupe d'un objet de référence par un grand nombre d'objets.

L'Algorithme 6.5 permet de couper un objet de référence O par un ensemble de petits objets $(a_i)_{1 \leq i \leq n}$ que l'on suppose disjoints. Au sein de la méthode améliorée de construction de la géométrie de la cellule, il assure les étapes (Alg. 6.3 L.9) et (Alg. 6.3 L.10). Cette méthode a été implémentée en Python dans le cadre de ces travaux de thèse sous le nom MAKECUTBYSHELLS, disponible à l'annexe §B.1.2.

Algorithme 6.5 Coupe d'un objet de référence par un grand nombre d'objets.

Entrée: L'objet de référence O.
Entrée: L'ensemble des objets disjoints $(a_i)_{1 \leq i \leq n}$ à ôter de O.
Sortie: L'objet coupé $T = O \setminus \cup_{1 \leq i \leq n} a_i$.

1: $t \leftarrow \partial O$.
2: **Pour** chaque objet a_i **faire**
3: $\quad s_i \leftarrow O \cap \partial a_i$.
4: $\quad t \leftarrow (t \setminus a_i) \cup s_i$.
5: **Fin Pour**
6: Construire le solide T associé à t.
7: **Renvoi** T.

Preuve de la correction de l'Algorithme 6.5:

Soit A et B deux solides, on va tout d'abord préciser la définition du solide $A \setminus B$. *Du point de vue strictement ensembliste et en utilisant la propriété de partition (6.3), on a :*

$$A \setminus B = ((A \setminus B) \cap \partial B) \sqcup \left((A \setminus B) \cap \complement B\right). \tag{6.11}$$

Ce qui se simplifie en :

$$x \in (A \setminus B) \cap \partial B \Leftrightarrow \begin{cases} x \in A \\ x \notin B \\ x \in \partial B \end{cases} \Leftrightarrow x \in A \cap \partial B,$$

et

$$x \in (A \setminus B) \cap \complement B \Leftrightarrow \begin{cases} x \in A \\ x \notin B \\ x \in \complement B \end{cases} \Leftrightarrow x \in A \cap \complement B.$$

En appliquant (6.9) à A et $\complement B$ *et compte tenu du fait que* $\partial \complement B = \partial B$, *il vient :*

$$\partial(A \cap \complement B) = \left(\partial A \cap \complement B\right) \sqcup (A \cap \partial B) \sqcup (\partial A \cap \partial B). \tag{6.12}$$

*On voit donc que l'*ensemble $A \setminus B$ *est composé d'un* solide $(A \cap \complement B)$ *et d'une* surface $(A \cap \partial B)$ *qui fait partie de la frontière de ce solide. Le solide correspondant à l'opération* $A \setminus B$ *est donc le solide* $A \cap \complement B$, *et on a :*

$$\partial(A \setminus B) = \left(\partial A \cap \complement B\right) \sqcup (A \cap \partial B) \sqcup (\partial A \cap \partial B). \tag{6.13}$$

Par un raisonnement similaire, l'ensemlbe $\partial A \setminus B$ *est composé d'une* surface $(\partial A \cap \complement B)$ *et d'une* courbe $(\partial A \cap \partial B)$, *d'où :*

$$\partial(A \setminus B) = (\partial A \setminus B) \sqcup (A \cap \partial B). \tag{6.14}$$

Appliquée aux objets O et a_0, *l'équation précédente permet d'assurer :*

$$\partial(O \setminus a_0) = (\partial O \setminus a_0) \sqcup s_0, \tag{6.15}$$

ce qui assure la correction de l'Algorithme 6.5 dans le cas où l'objet de référence O est coupé par un unique objet a_0. *Le cas général s'obtient naturellement par récurrence sur le nombre n d'objets disjoints à ôter de O.*

■

6.2.4 Gestion des points de tangence.

6.2.4.1 Origine des points de tangence.

Comme on l'a défini à la section §6.2.1, les inclusions présentes dans le domaine Ω sont disjointes les unes des autres. On peut donc définir une distance d_{min} telle que toute inclusion soit distante d'au moins d_{min} de ses voisines. En exigeant d'un mailleur qu'il génère des éléments de taille inférieure à d_{min}, on obtiendra nécessairement une discrétisation du domaine de travail Ω ou des cellules $\hat{K}$, sans préjugé de la dimension d'une telle discrétisation.

Un problème peut cependant survenir lors de la discrétisation des cellules $\hat{K}$. En effet, la séparation du domaine en cellules nécessite l'ajout à la géométrie du domaine de plans de découpes, les faces des zones homogènes $\left\{K_0, K_0^c\right\}$ de la cellule. La position de ces plans ne dépend que du nombre de cellules désirées à travers le domaine, et nullement de sa géométrie. Nombres d'inclusions sont donc coupées par les plans de découpe, et se retrouvent ainsi réparties sur plusieurs cellules. On crée ainsi artificiellement une complexité géométrique supplémentaire.

Généralement, la prise en charge de ces inclusions tronquées est assurée lors de la construction de la géométrie de la cellule. Si les sections d'inclusions présentes sur les parois des zones homogènes sont intégrées à la géométrie de la cellule, le mailleur en tiendra compte lors de la discrétisation des faces de la cellule. Le maillage fin $T_h(\hat{K})$ est ensuite obtenu en générant une discrétisation volumique cohérente avec cette discrétisation surfacique.

Un problème survient quand une inclusion et un plan n'ont pour intersection qu'un ensemble de points isolés. On dit dans ce cas que l'inclusion est *tangente* au plan et les points d'intersection sont appelés *points de tangence*. Ce cas limite empêche la génération automatique du maillage de la cellule, et ce quelle que soit la méthode utilisée pour assembler la géométrie de la cellule. Dans le premier cas (cf. Algorithme 6.1 §6.2.2), la méthode MAKEPARTITION n'arrive pas à assembler la cellule. Quant à la seconde méthode (cf. Algorithme 6.3 §6.2.3), si elle arrive bien à construire une cellule, *SALOME* ne parvient pas à la mailler. Il n'est pas possible en effet, au mailleur de tenir compte d'un point qui ne lui a pas été explicitement décrit dans la géométrie, ce que ne fait pas la méthode.

Comme on l'a écrit plus haut, la création de points de tangence ne dépend que du découpage choisi par l'utilisateur de la méthode (nombre de cellules, taux de sur-échantillonnage) et non du domaine. On peut donc envisager d'adapter le domaine au découpage, en déplaçant légèrement les inclusions tangentes. On résoudrait ainsi le problème, tout en conservant un milieu très proche à l'original. Cette implémentation de la méthode multi-échelle a cependant été développé dans l'optique de pouvoir réaliser des simulations extensives sur les matériaux cimentaires. Chaque milieu fera alors l'objet de plusieurs simulations, avec des découpages distincts. Adapter le milieu au cas par cas créera alors autant de domaines distincts, sur lesquels il sera difficile de comparer des résultats.

On a donc retenu une autre solution : ajouter explicitement les points de tangence à la géométrie des cellules.

6.2.4.2 Résolution du problème.

Pour une cellule donnée, et pour chaque paroi des zones homogènes, on suppose connu l'ensemble des points de tangence $\mathscr{P}$ à cette paroi. On cherche à intégrer cette information à la géométrie de la cellule, tout en minimisant les contraintes imposées aux parois.

Algorithme 6.6 Ajout d'un ensemble de points à un rectangle.

Entrée: Le rectangle O à corriger.
Entrée: Les points $\mathscr{P}$ à intégrer à O.
Sortie: L'ensemble T des segments courant sur le rectangle.
1: Initialiser T aves les arêtes du rectangle O.
2: **Pour** chaque point p de $\mathscr{P}$ **faire**
3: Déterminer $s_h(p)$ et $s_v(p)$ dans T.
4: Ajouter $\{s_h(p), s_v(p)\}$ à T.
5: **Fin Pour**
6: **Renvoi** T.

Au sein de *SALOME*, il n'est pas possible d'intégrer un point isolé à un solide. Pour palier à ce défaut, on procéde de la manière suivante, résumée à l'Algorithme 6.6 : à chaque point $p \in \mathscr{P}$, on associe deux segments du plan, un horizontal $s_h(p)$ et un vertical $s_v(p)$ se coupant en p. Ces segments s'étendent jusqu'à couper un segment de direction perpendiculaire ; c'est-à-dire couper un segment vertical pour $s_h(p)$ et un segment horizontal pour $s_v(p)$. Chaque point est ainsi placé au sein d'une découpe hiérarchique du milieu construit par les précédents points.

La Figure 6.6 présente une illustration de l'Algorithme 6.6. Cette méthode a été implémentée en Python dans le cadre de ces travaux de thèse sous le nom MAKECONFORMTOTANGENTOBJECTS, disponible à l'annexe §B.1.3.

Cette méthode de placement s'inspire de techniques développées pour les simulations astronomiques [35], où il est nécessaire de pouvoir traiter rapidement un très grand nombre de points. Normalement, il ne sera pas nécessaire de gérer autant de points, les points de tangence devant être, statistiquement, l'exception et non la règle. Cette méthode a cependant été retenue car elle limite la taille des segments générés. Les contraintes artificiellement ajoutées aux faces des cellules en sont limitées d'autant.

Remarque 6.4 *Il n'existe pas de méthode simple pour connaître,* via SALOME*, les points de tangences entre une inclusion de forme quelconque et un plan donné.*

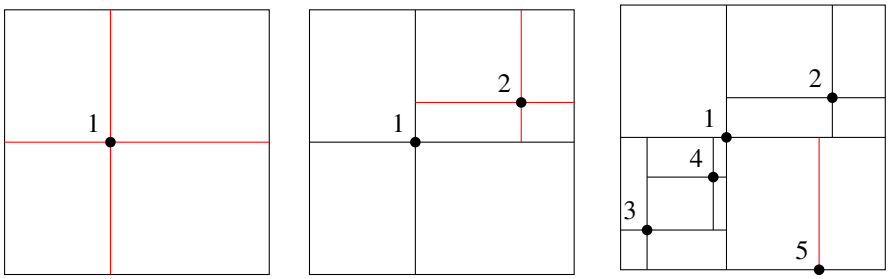


FIGURE 6.6 – Ajout de points à une face par l'Algorithme 6.6. On ajoute d'abord le premier point de tangence (à gauche), puis le second (au milieu), jusqu'à obtenir l'ensemble des points (à droite).

L'application de cet algorithme de correction a donc été restreint aux seules inclusions sphériques, pour lesquelles il est possible de calculer explicitement les éventuels points de tangence.

Remarque 6.5 *L'expérience montre qu'il est nécessaire d'introduire une certaine tolérance dans le calcul des points de tangence, et donc de traiter les inclusions très proches des parois de la cellule. On ajoute dans ces cas un point de* quasi-tangence*, i.e le point de la paroi le plus proche de l'inclusion. Cependant, cette tolérance ne résout pas tous les problèmes et certaines cellules restent très difficiles voire impossibles à discrétiser en l'état.*

6.2.5 Test de l'algorithme complet.

Algorithme 6.7 Génération des maillages de cellules $T_h(\hat{K})$.

1: Répartir les inclusions entre les cellules. ▷ cf. Algorithme 6.2.
2: **Pour** chaque cellule $\hat{K}$ **faire**
3: Créer $\left\{K_0, K_0^c\right\}$, les zones homogènes de la cellule.
4: Créer les inclusions $I \in \mathscr{I}^K$ de la cellule.
5: Ajouter les points de tangences à $\left\{K_0, K_0^c\right\}$. ▷ cf. Algorithme 6.6.
6: Assembler la cellule géométrique $\hat{K}$. ▷ cf. Algorithme 6.3.

7: Marquer les groupes d'inclusions au sein de la cellule.
8: Marquer les groupes de bords au sein de la cellule.

9: Mailler la cellule $\hat{K}$.
10: Marquer les familles d'éléments à partir des groupes correspondants.
11: Exporter le maillage au format MED.
12: **Fin Pour**

On reprend l'Exemple 6.1 présenté à la section §6.2.2.6.

La Figure 6.7 présente le temps nécessaire afin de générer les géométries des domaines Ω_n, pour $0 \leq n \leq 9$. Le domaine Ω_n comporte $N = (n+1)^3$ inclusions.

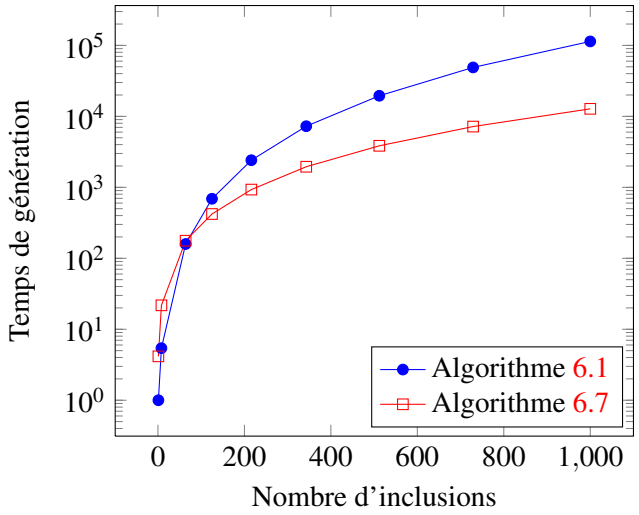


FIGURE 6.7 – Temps nécessaires à la génération de la géométrie du milieu Ω_n, en fonction du nombre d'inclusions $N = (n+1)^3$. Les temps présentés ont été moyennés sur 10 construction et on a choisi le temps moyen nécessaire à l'Algorithme 6.1 pour générer Ω_0 comme unité de référence.

La méthode native *SALOME* obtient de meilleurs résultats pour un petit nombre d'inclusions. Très rapidement cependant, la seconde méthode devient compétitive, jusqu'à obtenir un temps de génération sensiblement identique pour $N = 64$. Au-delà de cette valeur, l'Algorithme 6.7 est de plus en plus efficace. Pour l'exemple le plus complexe, le milieu Ω_9 à 1000 inclusions, le facteur de gain de temps atteint presque 9. Même s'il est peu probable qu'une cellule contienne un nombre aussi important d'inclusions, ces bons résultats confortent l'intérêt de travailler en détail la génération des géométries, en lieu et place de la simple utilisation des routines natives *SALOME*.

6.3 Génération de maillages coïncidents.

6.3.1 Problématiques et idées de solutions.

6.3.1.1 Besoin en maillages coïncidents.

Selon le principe de la méthode multi-échelle, une fois le milieu divisé en cellules, celles-ci sont indépendantes. Le seul lien d'une cellule à l'autre est la paroi des macroéléments sous-jacents. Même s'il s'agit d'une seule réalité physique, il n'est pas indispensable d'assurer que le maillage sur ces plans soit identique d'une

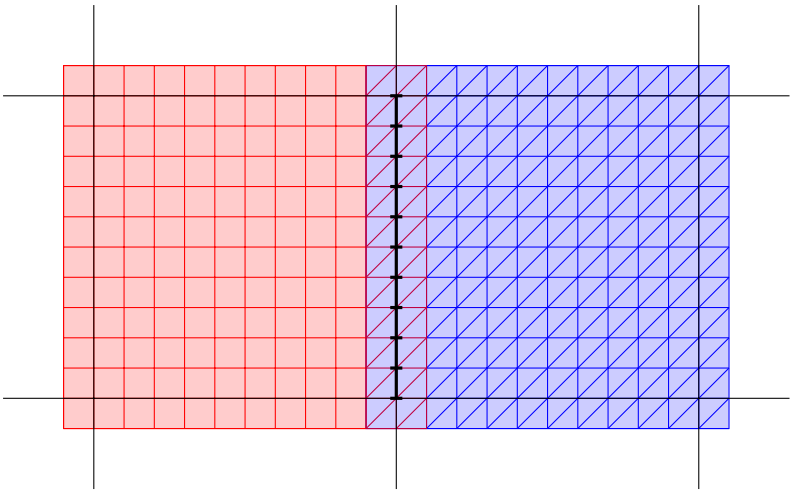


FIGURE 6.8 – Maillages coïncidents : on construit les cellules $\hat{K}^-$ (en rouge) et $\hat{K}^+$ (en bleu) de deux macroéléments voisins K^- et K^+. Les discrétisations $T_h(\hat{K}^-)$ et $T_h(\hat{K}^+)$ sont distinctes, et certaines zones du milieu ont donc deux discrétisations. Cependant la discrétisation de $\partial K^- \cap \partial K^+$ (en noir) est la même dans les deux maillages.

cellule à l'autre.

Ainsi les méthodes multi-échelles Q_1/*VF9* et Q_1/*VFDiam*, présentées au chapitre §3, n'en ont pas besoin pour fonctionner. Tout au plus l'unicité de ce maillage permet-il d'obtenir un maillage continu lors des opérations de post-traitements (§3.4), comme le recalcul de la solution fine $C_{H,h}$ sur une portion du domaine Ω. Le recollement des solutions en un seul maillage est cependant plus affaire de confort que d'intérêt. Il est en effet tout aussi facile, sinon plus, d'afficher huit petits maillages au lieu d'un seul gros, ou de calculer les taux d'erreur cellule par cellule plutôt que sur le domaine complet.

Cependant, si l'on souhaite utiliser un schéma de Galerkin discontinu à l'échelle grossière, par exemple dans le cadre de la méthode multi-échelle GD/*VFDiam* (cf. §3.3.2), la donne change considérablement. En effet, ce schéma demande de calculer de nombreux termes croisés d'une cellule à l'autre, c'est-à-dire des quantités de la forme :

$$A = \int_{\partial K} a^- b^+ \tag{6.16}$$

où $\partial K = \partial K^- \cap \partial K^+$ représente la frontière commune aux deux cellules voisines $\hat{K}^-$ et $\hat{K}^+$, que on a pu en représenter à la Figure 6.8.

a^- et b^+ sont des fonctions définies respectivement sur $\hat{K}^-$ et $\hat{K}^+$. Dans les méthodes de Galerkin Discontinues, il s'agit de combinaisons linéaires de la solution,

de son gradient et de son flux. Ces trois quantités sont calculées lors de la résolution des problèmes de cellules, et donc discrétisées au niveau fin.

Si l'on dispose d'une discrétisation fine $T_h(\partial K)$ de ∂K, on peut aisément calculer une approximation $\tilde{A}$ de A :

$$\tilde{A} = \sum_{f \in T_h(\partial K)} |f| a_h^{-}(f) b_h^{+}(f) \tag{6.17}$$

Si on ne dispose pas d'une unique discrétisation de ∂K, il existe d'autres possibilités pour calculer une approximation du terme A. On peut par exemple construire une méthode d'Éléments Finis *Mortiers* [85, 139, 173] sur ∂K à partir des maillages des cellules adjacentes. On peut également projeter, ou interpoler, les valeurs d'une discrétisation sur l'autre.

Ces méthodes risquent cependant de s'avérer très complexes à mettre en œuvre, puisque les maillages fins des cellules ne sont pas réguliers. La mise en relation d'un maillage à un autre est en effet une tâche algorithmiquement lourde à effectuer.

6.3.1.2 Utilisation de mailleurs déterministes.

Pour obtenir un maillage coïncident sur les frontières des macroéléments, une première solution est de recourir à des mailleurs déterministes. Comme leurs noms l'indique, pour le même jeu de paramètres (domaine, pas du maillage, etc), ces mailleurs construient un unique maillage, quel que soit le nombre de fois où l'on fait appel à eux. En appliquant spécifiquement ce mailleur à la frontière ∂K du macroélément, celle-ci sera discrétisée de manière identique dans les deux cellules concernées.

Deux obstacles s'opposent cependant à l'utilisation des mailleurs déterministes. En premier lieu, il en existe très peu : parmi les mailleurs $2D$ fournis avec *SALOME*, seul NETGEN est déterministe.

Le second problème survint après la discrétisation de la frontière, une fois $T_h(\partial K)$ obtenu. Le mailleur $3D$ va alors discrétiser la cellule, afin d'obtenir $T_h(\hat{K})$, en tenant compte de $T_h(\partial K)$. Cependant ces mailleurs $3D$ modifient généralement le maillage $2D$ au cours de leur travail. Ces modifications sont généralement minimes : points déplacés, mailles divisées ou fusionnées, etc. mais suffisent à rendre chaque maillage de frontière $T_h(\partial K)$ dépendant de la cellule $\hat{K}$.

6.3.1.3 Méthode des jeux successifs.

On a ensuite envisagé de mailler les cellules $\hat{K}$ dans un ordre précis. Il s'agit en fait de mailler en premier lieu une unique cellule $\hat{K}^{-}$, par exemple celle d'indice $(0,0,0)$, qui servira de référence. Ensuite, on travaille sur les cellules $\hat{K}^{+}$ voisines de $\hat{K}^{-}$, en tenant compte de $T_h(\hat{K}^{-})$. On projette tout d'abord la discrétisation de ∂K provenant de $T_h(\hat{K}^{-})$ sur les parois correspondantes de $\hat{K}^{+}$, puis on maille le reste de la cellule, parois et solide, à partir de ce début de maillage.

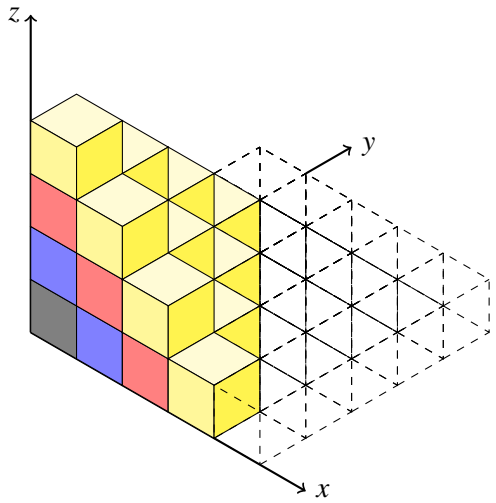


FIGURE 6.9 – Maillages coïncidents par la méthode des jeux successifs. Pour obtenir des maillages fins coïncidents, les cellules sont traitées de proche en proche à partir de la cellule $(0,0,0)$ (en gris). Le maillage de chaque cellule est construit en tenant compte des surfaces maillées dans les jeux précédents. On représente ici les cellules par leurs macroéléments (cubes) sous-jacents, les jeux successifs apparaissant de différentes couleurs.

Cette méthode de *projection* permet de verrouiller les sous-maillages correspondants aux faces : le mailleur 3*D* ne peut donc les modifier lors de son travail. De proche en proche, on obtient ainsi un ensemble de maillages coïncidents sur les cellules du domaine.

Cette solution a cependant été écartée pour plusieurs raisons, notamment liées au parallélisme de la méthode multi-échelle. En premier lieu, compte tenu de la manière dont *SALOME* fonctionne, cette méthode a besoin des objets GEOM et MESH d'une cellule $\hat{K}^-$, ainsi que de l'étude associée, pour pouvoir mailler les voisins de $\hat{K}^-$. Projeter une partie d'un maillage sur un autre nécessite en effet de disposer des deux objets géométriques et des deux sous-maillages concernés. À chaque étape, il est donc nécessaire de stocker l'étude complète correspondante, avant de la transmettre au processeur chargé du travail sur $\hat{K}^+$.

En second lieu, cette solution limite le nombre de cellules que l'on peut discrétiser en même temps, puisqu'elle établit un *ordre* de traitement. Au mieux, on peut espérer traiter les cellules par jeu $\{X+Y+Z=cst\}$, comme on l'a illustré à la Figure 6.9. Cette limitation n'est donc un véritable problème que si le nombre de processeurs alloués à la discrétisation des cellules est important. Dans le cas contraire, le travail peut être efficacement réparti, même si on ne dispose pas de la totale indépendance des cellules que l'on pouvait normalement espérer.

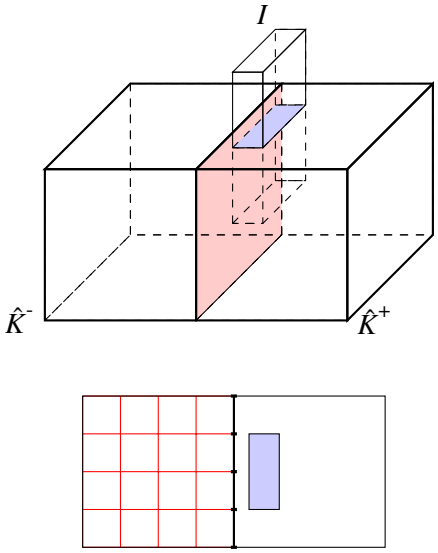


FIGURE 6.10 – Maillages coïncidents par la méthode des jeux successifs.
En haut : deux cellules voisines $\hat{K}^-$ (à gauche) et $\hat{K}^+$ (à droite), pour lesquelles on a pris $\rho = 0$, et une inclusion I de $\hat{K}^+$ située près de $\partial K = \partial K^- \cap \partial K^+$ (en rouge).
En bas : discrétiser $\hat{K}^+$ sera difficile car la discrétisation $T_h(\partial K)$ de ∂K, héritée de $T_h(\hat{K}^-)$, n'est pas adaptée à la présence d'une inclusion si proche.

Enfin, cette méthode impose des contraintes aux mailleurs qui pourraient se révéler incompatibles avec la géométrie de la cellule. En effet, par ce biais, jusqu'à trois des six faces d'un macroélément ont hérité un maillage de précédentes discrétisations. Or, si les nouvelles faces présentent une géométrie beaucoup plus complexe que les faces projetées, il n'est pas sûr que les mailleurs puissent discrétiser la cellule.

Considérons par exemple le cas décrit à la Figure 6.10, dans lequel on confond cellule et macroélément ($\rho = 0$) pour simplifier. Une unique inclusion I coupe la face à mailler d'une cellule $\hat{K}^+$ (à droite), et cette coupe est très proche d'une autre face, celle-ci héritée d'une précédente cellule $\hat{K}^-$ (à gauche). La face projetée ne présentant aucune inclusion, le mailleur n'a aucune raison de la discrétiser finement lors de son travail sur $\hat{K}^-$. Le maillage $T_h(\hat{K}^-)$ (en bas, à gauche) est donc très grossier. Lors de son travail sur $\hat{K}^+$, le mailleur doit donc discrétiser l'espace entre $\partial\hat{K}^+$ et l'inclusion I, qui est très petit, en partant d'un maillage grossier qu'il ne peut modifier, puisque le maillage projeté depuis $T_h(\hat{K}^-)$ est verrouillé. Dans ces conditions, rien ne garantit que le mailleur arrive à générer une discrétisation $T_h(\hat{K}^+)$ convenable.

6.3.1.4 Méthode du maillage de peau.

Finalement, on a choisi de mailler séparément les frontières de macroéléments et les cellules. Dans un premier temps, on construit un objet *SALOME* regroupant les plans de découpes du milieu, c'est-à-dire l'ensemble des frontières de macroéléments ∂K et les inclusions les coupant. Cet objet est alors maillé puis stocké en mémoire, géométrie comprise. Même dans le cas de très grands domaines, le nombre de mailles nécessaires à la discrétisation d'un tel objet reste raisonnable, puisqu'il ne s'agit que d'un maillage $2D$.

Lors du travail sur une cellule, cette *étude de référence* est chargée en mémoire. La construction de la géométrie de la cellule reste inchangée par rapport aux méthodes précédentes (cf. Algorithme 6.7 §6.2.3), la seule différence intervenant lors de la génération du maillage proprement dit. Comme dans la méthode décrite à la section précédente, chaque sous-objet *SALOME* appartenant à la paroi du macroélement est alors identifié à son équivalent sur la géométrie de référence, comme l'illustre la Figure 6.11. Le maillage de référence est alors projeté sur le nouvel objet, puis on demande au mailleur global de terminer la discrétisation de la cellule.

6.3.2 Description algorithmique.

6.3.2.1 Génération de l'étude de référence.

L'Algorithme 6.8 décrit la construction de la géométrie de référence et de son maillage $2D$. La première étape Alg. 6.8 L.1 consiste à déterminer l'ensemble des inclusions $\mathscr{I}_{MC}$ ayant une intersection non nulle avec les parois des macroéléments. On peut calculer cet ensemble lors de la répartion des inclusions au sein des cellules, les ensembles $\mathscr{I}_{MC}$ et $\mathscr{I}_{MC}^{K}$ étant liés par la relation suivante :

$$\mathscr{I}_{MC} = \bigcup_{\hat{K}} \mathscr{I}_{MC}^{K} \tag{6.18}$$

La suite de l'algorithme ne récèle pas de difficultés particulières. L'assemblage de l'objet de référence utilise la routine native MAKEPARTITION et non pas une implémentation adaptée comme on le propose à la section §6.2.3. En effet, on travaille ici en $2D$, et le coût de MAKEPARTITION reste donc raisonnable.

6.3.2.2 Algorithme complet de génération des maillages coïncidents.

L'Algorithme 6.9 décrit la génération complète de l'ensemble des maillages coïncidents de cellule. Une fois l'étude *SALOME* de référence construit *via* l'Algorithme 6.8, le travail sur chaque cellule reprend les étapes décrites à l'Algorithme 6.7 : construction des zones homogènes et des inclusions de la cellule, assemblage de la cellule, puis marquage des groupes d'éléments.

Avant de générer le maillage, chaque sous-objet appartenant à la paroi du macroélement est lié à son alter-ego de l'objet de référence. Le mailleur discrétise

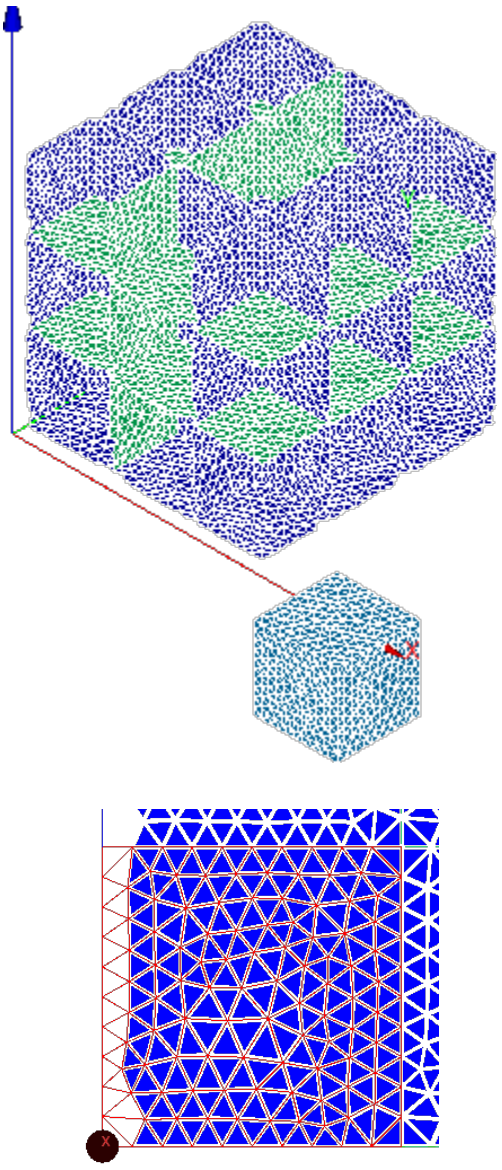


FIGURE 6.11 – Maillages coïncidents par maillage de peau : captures d'écran de la plate-forme *SALOME*.
En haut : le maillage de référence pour le découpage $(\mathscr{D},\rho) = (\{3,3,3\},0)$ et le maillage de la cellule $\hat{K} = (2,0,0)$. La frontière $\partial\Omega$ du milieu Ω n'est pas représentée afin de montrer la discrétisation de l'ensemble des frontières ∂K.
En bas : les discrétisations de ∂K issues du maillage de référence (en bleu) et de $T_h(\hat{K})$ (en rouge) sont identiques.

Algorithme 6.8 Génération du maillage $2D$ de référence.

Entrée: Le découpage du milieu.
Entrée: La liste $\mathscr{I}$ des inclusions.
Sortie: Une étude *SALOME* de référence.
1: Déterminer $\mathscr{I}_{MC}$.

2: Créer les plans de coupe du domaine Ω.
3: Créer les inclusions $\mathscr{I}_{MC}$.
4: Assembler la géométrie de référence.

5: Mailler la géométrie de référence.

6: Sauvegarder l'étude *SALOME* de référence.

ensuite le domaine en tenant compte des éléments déjà existant, en l'occurence la discrétisation $T_h(\partial K)$ des frontières du macroélément K.

6.4 Évolution des maillages avec la dégradation du milieu.

6.4.1 Présentation de la méthode.

La présence d'une solution agressive modifie considérablement les propriétés des matériaux cimentaires (cf §3.6.2). En particulier, certaines espèces chimiques peuvent disparaître, modifiant ainsi le domaine de travail. On modélise dans cette section un processus de dégradation par le biais d'une succession de domaines $(\Omega^i)_{0\leq i\leq n}$ représentant le milieu à différents stades de dégradation. On suppose ces domaines explicitement connus, ce qui signifie que que le milieu à un instant donné ne dépend pas de simulations effectuées sur le domaine à des instants antérieurs. On considère par convention que le domaine Ω^0 est sain et que Ω^n est le plus dégradé.

Compte tenu du fait que le phénomène de dégradation ne peut que rendre le milieu plus diffusif, les domaines dégradés vérifient la propriété d'irréversibilité suivante :

$$\forall 0 \leq i < n \qquad\qquad \Omega^i \subset \Omega^{i+1} \tag{6.19}$$

Afin de résoudre le problème de diffusion sur les domaines $(\Omega^i)_{0\leq i\leq n}$, il est nécessaire de déterminer leurs discrétisations $(T_h(\Omega^i))_{0\leq i\leq n}$. Il est bien sûr possible de générer indépendamment chacune de ces discrétisations, mais cette tâche est d'autant plus coûteuse que le nombre de domaines est important. De plus, cette approche directe n'utilise pas la relation qui lient les différents domaines dégradés.

Les domaines dégradés étant inclus les uns dans les autres, il paraît naturel de créer leurs discrétisations de proche en proche. Cependant, pour des raisons tech-

Algorithme 6.9 Génération des maillages de cellules $T_h(\hat{K})$ coincidents.

1: Répartir les inclusions entre les cellules. ▷ cf. Algorithme 6.2.
2: Construire l'étude *SALOME* de référence. ▷ cf. Algorithme 6.8.
3: **Pour** chaque cellule $\hat{K}$ **faire**
4: Charger l'étude *SALOME* de référence.
5: Créer $\left\{K_0, K_0^c\right\}$, les zones homogènes de la cellule.
6: Créer les inclusions $I \in \mathscr{I}^K$ de la cellule.
7: Ajouter les points de tangences à $\left\{K_0, K_0^c\right\}$. ▷ cf. Algorithme 6.6.
8: Assembler la cellule géométrique $\hat{K}$. ▷ cf. Algorithme 6.3.

9: Marquer les groupes d'inclusions au sein de la cellule.
10: Marquer les groupes de bords au sein de la cellule.

11: Lier ∂K à son équivalent sur l'objet de référence.
12: Mailler la cellule $\hat{K}$, en tenant compte des éléments déjà existants.
13: Marquer les familles d'éléments à partir des groupes correspondants.
14: Exporter le maillage au format MED.
15: Fermer l'étude *SALOME* de référence.
16: **Fin Pour**

niques, il n'est pas facile d'ajouter des éléments à un maillage. Il est en revanche relativement aisée d'en ôter, et c'est dans cette optique que le problème a été abordé *à rebours*.

Pour réaliser le jeu de domaines dégradés, on discrétise tout d'abord Ω^n, le dernier milieu, le plus dégradé, celui donc qui a le domaine de travail le plus étendu. Cette discrétisation tient compte des diverses géométries du milieu, de sorte que toutes les zones à dégrader soient identifiables. Dans cette optique, un *groupe* rassemble des inclusions aux caractéristiques physiques communes ; c'est la définition de la section §6.2.2.4 ; et qui subiront de plus le processus de dégradation ensemble. Les domaines dégradés sont ensuite créés de proche en proche, en suivant le processus de dégradation à rebours. Au fur et à mesure que l'on remonte le processus de dégradation physique, les zones dégradées redeviennent imperméables et on supprime les éléments correspondants du maillage.

À chaque étape de dégradation, une fois construit le maillage $T_h(\Omega^i)$, il est nécessaire de déterminer quelles faces du maillage décrivent les frontières du milieu Ω^i. La construction de $T_h(\partial\Omega^i)$ ne suffit cependant pas : on doit également répartir ces faces frontières en plusieurs groupes, en fonction de leur origine.

Il s'agit en fait de distinguer les anciennes faces frontières, c'est-à-dire appartenant également à $T_h(\partial\Omega^{i+1})$, des nouvelles, celles issues de la suppression d'éléments de $T_h(\Omega^{i+1})$. En effet, ces faces discrétisent une interface particulière, et on peut vouloir, lors des simulations de diffusion, leurs appliquer des conditions aux

limites en conséquence. À charge donc, lors de la constuction des maillages, de tenir compte de ce besoin et de distinguer précisement les contributions de chaque dégradation à la frontière du domaine.

6.4.2 Description algorithmique.

Pour tout $0 \leq i < n$, on nomme *zones de dégradation* et on note ω^{i+1} le domaine libéré par la disparition de composantes entre les étapes i et $i+1$. $(\Omega^i)_{0\leq i\leq n}$ et $(\omega^i)_{0<i\leq n}$ vérifient donc les relations suivantes :

$$\forall 0 \leq i < n, \qquad \Omega^i = \Omega^{i+1} \setminus \omega^i \tag{6.20}$$

$$\forall 0 < i \leq n, \qquad \omega^i \subset \Omega^i \tag{6.21}$$

$$\forall 0 < i < j \leq n, \qquad \omega^i \cap \omega^j = \emptyset \tag{6.22}$$

Dans tous ce qui suit, on notera A_h la discrétisation $T_h(A)$ de l'objet A, afin d'alléger les notations.

On suppose que les zones de dégradation $(\omega^i)_{0<i\leq n}$ admettent chacune une discrétisation ω_h^i. On suppose en outre disposer de Ω_h^n, une discrétisation du milieu entièrement dégradé Ω^n tenant compte des discrétisations $(T_h(\omega^i))_{0<i\leq n}$, i.e :

$$\forall 0 < i \leq n \qquad \omega_h^i \subset \Omega_h^n \tag{6.23}$$

On note $\Gamma^i = \partial\Omega^i$ et $\gamma^i = \partial\omega^i$. On suppose que Γ^n est composé des ensembles disjoints $(\Gamma^{n,j})_{0\leq j\leq q^n}$. Ces ensembles représentent, par exemple, les supports de différentes conditions aux limites du problème de diffusion à résoudre. Ils sont compatibles avec la discrétisation Γ_h^n de Γ^n, au sens où l'on suppose que :

$$\forall 0 < j \leq q^n \qquad \Gamma_h^{n,j} \subset \Gamma_h^n. \tag{6.24}$$

Définition 6.9 *Pour $0 < i \leq n$, on définit plusieurs ensembles de faces, qui permettent de construire la partition de Γ_h^i :*

$$\forall 0 \leq j \leq q^i \qquad \delta_h^{i,j} = \Gamma_h^{i,j} \cap \gamma_h^i \tag{6.25}$$

$$\forall 0 \leq j \leq q^i \qquad \Sigma_h^{i,j} = \Gamma_h^{i,j} \setminus \delta_h^{i,j} \tag{6.26}$$

$$\sigma_h^i = \gamma_h^i \setminus \left\{ \cup_{0\leq j\leq q^i} \delta_h^{i,j} \right\} \tag{6.27}$$

L'Algorithme 6.10 présente la construction du maillage Ω_h^i à partir du maillage Ω_h^{i+1}. Il construit la discrétisation de Γ^{i+1} sous la forme de frontières disjointes $(\Gamma_h^{i,j})_{0\leq j\leq q^i}$ en mettant à jour les frontières déjà existantes et en identifiant la nouvelle frontière. On présente à la Figure 6.12, un jeu de maillages générés par cette méthode.

Algorithme 6.10 Discrétisation d'un domaine Ω^i à partir de la discrétisation Ω_h^{i+1} du domaine dégradé Ω^{i+1}.

Entrée: Les maillages Ω_h^{i+1} et ω_h^{i+1}.
Entrée: Les maillages $(\Gamma_h^{i+1,j})_{0\leq j\leq q^{i+1}}$ et γ_h^{i+1}.
Sortie: Les maillages Ω_h^i et $(\Gamma_h^{i,j})_{0\leq j\leq q^i}$.

$\Omega_h^i \leftarrow \Omega_h^{i+1} \setminus \omega_h^{i+1}$.
$\sigma_h^i \leftarrow \gamma_h^{i+1}$.

Pour $0 \leq j \leq q^{i+1}$ **faire**
    $\delta_h^{i+1,j} \leftarrow \Gamma_h^{i+1,j} \cap \gamma_h^{i+1}$.
    $\Sigma_h^{i+1,j} \leftarrow \Gamma_h^{i+1,j} \setminus \delta_h^{i+1,j}$.
    $\sigma_h^{i+1} \leftarrow \sigma_h^{i+1} \setminus \delta_h^{i+1,j}$.

    $\Gamma_h^{i,j} \leftarrow \Sigma_h^{i+1,j}$.
Fin Pour
$q^i \leftarrow q^{i+1} + 1$.
$\Gamma_h^{i,q^i} \leftarrow \sigma_h^{i+1}$.

Preuve de la correction de l'Algorithme 6.10:

On va montrer que les discrétisations construites par l'Algorithme 6.10 permettent de décrire correctement le domaine Ω^i et ses frontières. Etant donné la définition des domaines de dégradation (6.20) et de leurs discrétisations (6.23), il est clair que Ω_h^i est bien une discrétisation de Ω^i.

Reste donc à montrer que les ensembles de faces correspondent. On rappelle tout d'abord une propriété fondamentale de la discrétisation $(\partial A)_h$ de la frontière ∂A d'un objet A : un élément de $(\partial A)_h$ est la face d'un unique élément de A_h. Toute face appartenant à deux éléments distincts de A_h est donc intérieure à A.

$$(\partial A)_h \equiv \{f \text{ face de } A_h,\ \exists! k \in A_h,\ f \in \partial k\} \tag{6.28}$$

Si les ensembles $(\Gamma_h^{i+1,j})_{0\leq j\leq q^{i+1}}$ sont deux à deux disjoints, alors les ensembles $(\Sigma_h^{i+1,j})_{0\leq j\leq q^{i+1}}$ et σ_h^{i+1} sont deux à deux disjoints par construction.

Pour tout $0 \leq j \leq q^{i+1}$, soit f une face de $\Sigma_h^{i+1,j}$, alors f appartient à $\Gamma_h^{i+1,j}$. Il existe donc un unique élément k de $\Omega_h^{i+1,j}$ tel que $f \in \partial k$. Or $f \notin \gamma_h^{i+1}$, donc nécessairement $k \notin \omega_h^{i+1}$. k est donc un élément de Ω_h^i. Puisque Ω_h^i est inclus dans Ω_h^{i+1}, k est l'unique élément de Ω_h^i contenant f. Donc f appartient à Γ_h^i.

Soit f une face de σ_h^{i+1}, alors f n'appartient pas à Γ_h^{i+1}. Il existe donc exactement deux éléments disctincts k^+ et k^- de Ω_h^{i+1} contenant f. Un seul de ces deux

éléments, par exemple k^-, appartient à ω_h^{i+1} car f appartient à γ_h^{i+1}. k^+ est alors l'unique élément de Ω_h^i contenant f. Donc f appartient à Γ_h^i.

Soit f une face de Γ_h^i, alors il existe un unique élément k de Ω_h^i contenant f.

Si il existe $0 \leq j \leq q^{i+1}$ tel que f appartienne à $\Gamma_h^{i+1,j}$, alors k est unique dans Ω_h^{i+1}. Or k n'est pas un élément de ω_h^{i+1}, par construction de Ω_h^i. Donc f n'appartient pas à γ_h^{i+1}. D'après (6.26), f appartient donc à $\Sigma_h^{i+1,j}$.

Sinon, il existe un second élément k^+ dans Ω_h^{i+1} contenant f. k^+ appartient nécessairement à ω_h^{i+1}, puisqu'il n'appartient pas à Ω_h^i. Donc f appartient à γ_h^{i+1}. D'après (6.27), f appartient donc à σ_h^{i+1}.

On vient de montrer que les ensembles $(\Sigma_h^{i+1,j})_{0\leq j\leq q^{i+1}}$ et σ_h^{i+1} réalisent une partition de Γ_h^i, ce qui prouve la correction de l'algorithme.

■

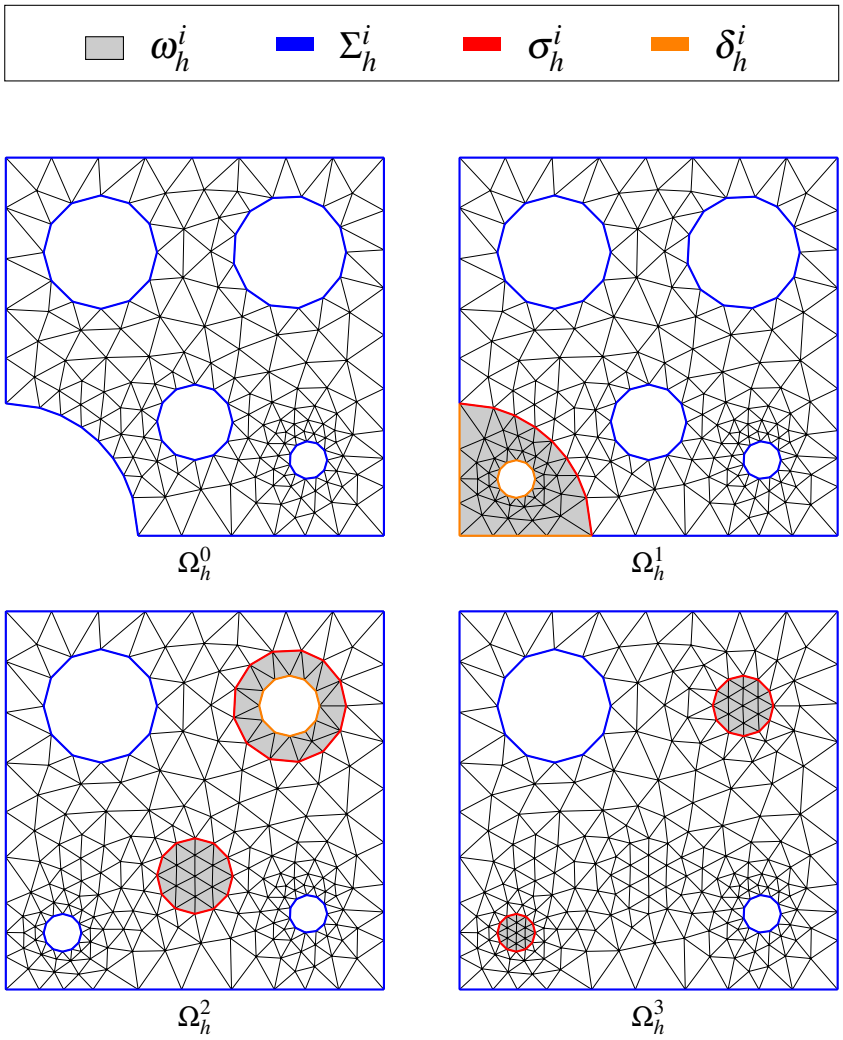


FIGURE 6.12 – Adaptation de la discrétisation d'un milieu au cours de sa dégradation. À partir de la discrétisation Ω_h^3, correspondant au milieu le plus dégradé, l'Algorithme 6.10 construit les discrétisations Ω_h^2, Ω_h^1 et Ω_h^0 en suivant le processus de dégradation à rebours. On a fait ici apparaître en couleurs les différents ensembles utilisés par l'Algorithme 6.10.

Chapitre 7

Mise en œuvre dans le cas $3D$: simulations.

Introduction.

Dans les deux précédents chapitres, on a présenté la chaîne de calcul multi-échelle *SALOME-MPCube* (§5) et les méthodes mises en œuvre afin de construire des maillages adaptés aux matériaux cimentaires (§6). On a de plus effectué des tests de l'implémentation des méthodes multi-échelles, qui ont été présentés à la section §5.2.

Ce chapitre présente des simulations effectuées par le biais de la chaîne de calcul *SALOME-MPCube* complète. Il s'agit d'exemples particuliers du problème de diffusion général que l'on rappelle ici :

$$\left\{\begin{array}{rcll} -\nabla\cdot(D\nabla C) & = & f & \text{dans } \Omega, \\ C & = & g_D & \text{sur } \Gamma_D, \\ D\nabla C\cdot\mathbf{n} & = & g_N & \text{sur } \Gamma_N, \end{array}\right. \tag{7.1}$$

où (Γ_D,Γ_N) forme une partition de $\Gamma=\partial\Omega$ la frontière du domaine de travail Ω. $\mathbf{n}$ designe le vecteur normal unitaire extérieure au domaine. f, g_D et g_N sont respectivement le terme source, la condition aux limites de Dirichlet et la condition aux limites de Neumann.

Ce chapitre se décompose en deux parties. On présente en premier lieu des exemples théoriques où la solution C du problème est connue, à commencer par une série de problèmes en milieu homogène qui permettent de vérifier l'implémentation du couplage entre les parties locale (problèmes de cellules, calcul des matrices locales) et globale (méthodes d'Éléments Finis et de Galerkin discontinue) de la chaîne de calcul (§7.1.1). Un exemple de milieu périodique est étudié à la section suivante §7.1.2. La section §7.1.3 présente la réponse de la chaîne de calcul à un découpage fin du domaine de travail Ω, quand le maillage grossier $T_H(\Omega)$ comporte un grand nombre de macro-éléments.

Dans une seconde partie, on présente des simulations effectuées sur des exemples de matériaux cimentaires, comme un échantillon de mortier (§7.2.1), une pâte cimentaire périodique (§7.2.2) et un échantillon aléatoire de pâte cimentaire (§7.2.3). Ce dernier exemple en particulier est représentatif des simulations que l'on souhaite réaliser sur les matériaux cimentaires : plusieurs dizaines de milliers d'inclusions, des sauts de diffusivité d'un facteur 1×10^{11} et une discrétisation du domaine Ω en près de 100 millions de tétrahèdres.

7.1 Exemples théoriques.

7.1.1 Qualification de la chaîne de calcul *SALOME-MPCube*.

Au chapitre §5, l'implémentation de chaque partie de la chaîne de calcul *SALOME-MPCube* a été qualifiée séparemment. La partie locale de la chaîne (résolution des problèmes de cellules, calcul des matrices locales) a ainsi été testée par le biais des Exemples 5.1 et 5.2, présentés à la section §5.2.2.5. À la section §5.2.3.2, les Exemples 5.3 à 5.8 valident l'implémentation des méthodes d'Éléments Finis et de Galerkin discontinue en dimension 2 et 3.

On présente maintenant plusieurs exemples du problème de diffusion (7.1) permettant d'assurer que le couplage entre les parties locale et globale de la chaîne de calcul *SALOME-MPCube* a été correctement implémenté.

Exemple 7.1 *On considère dans cet exemple le domaine* $\Omega =]0,1[^2$ *où l'on pose :*

$$D = \begin{bmatrix} a & 0 \\ 0 & 4a \end{bmatrix}$$

où a est un réel strictement positif. On choisit ici $a = 2$.

Dans ce cas, les fonctions $(\Phi_K^i)_{1 \leq i \leq n}$ *sont explicitement connues car elles correspondent aux fonctions d'Éléments Finis classiques* Q_1*, et ce pour toute valeur de* ρ *:*

$$\Phi_K^1 = (1-x)(1-y) \qquad \Phi_K^3 = (1-x)y$$
$$\Phi_K^2 = x(1-y) \qquad \Phi_K^4 = xy$$

On étudie trois problèmes de diffusion distincts sur le domaine Ω, inspirés des Exemples 5.3 à 5.5.

Exemple 7.1.1 *On considère dans cet exemple que :*

$$\begin{aligned} f &= 0, \\ \Gamma_D &= \partial\Omega, \\ g_D &= \sin(2q\pi x)\sinh(q\pi y), \end{aligned}$$

où q est un réel strictement positif. On choisit ici $q = 3/4$.

La solution du problème est alors :

$$C = \sin(2q\pi x)\sinh(q\pi y).$$

Exemple 7.1.2 *On note* $\Gamma_{y,m}$ *et* $\Gamma_{y,p}$ *les faces du domaine* Ω *pour lesquelles on a* $\{y = 0\}$ *et* $\{y = 1\}$ *respectivement. On considère dans cet exemple que :*

$$\begin{aligned} f &= 0, \\ \Gamma_N &= \Gamma_{y,m} \sqcup \Gamma_{y,p}, \\ g_N &= \frac{\partial}{\partial \mathbf{n}} \left(\sin(2q\pi x)\sinh(q\pi y)\right), \\ \Gamma_D &= \partial\Omega \setminus \Gamma_N, \\ g_D &= \sin(2q\pi x)\sinh(q\pi y), \end{aligned}$$

où q a été défini à l'exemple précédent.

La solution du problème est alors :

$$C = \sin(2q\pi x)\sinh(q\pi y)$$

Exemple 7.1.3 *On considère dans cet exemple que :*

$$\begin{aligned} f &= 8aq^2\pi^2 \sin(2q\pi x)\sin(q\pi y), \\ \Gamma_D &= \partial\Omega, \\ g_D &= \sin(2q\pi x)\sin(q\pi y) \end{aligned}$$

La solution du problème est alors :

$$C = \sin(2q\pi x)\sin(q\pi y).$$

On résout ces problèmes par les deux méthodes multi-échelles Q_1/*VFDiam* et GD/*VFDiam*, en suivant les étapes de la chaîne de calcul *SALOME-MPCube* présentée au chapitre §5.

On discrétise tout d'abord le domaine Ω en un maillage $T_H(\Omega)$ composé de macroéléments carrés de longueur H. Une fois fixé le taux de sur-échantillonage ρ, chaque cellule $\hat{K}$ est construite puis discrétisée *via SALOME* en un maillage $T_h(\hat{K})$ de pas $h = \alpha H$, avec $\alpha = 1/20$.

On traite ensuite chaque cellule $\hat{K}$ du domaine : résolution des problèmes locaux, détermination de la base locale $(\Phi_K^i)_{1\leq i\leq n}$ et enfin calcul des matrices locales. On rappelle que cette étape ne dépend que du milieu Ω où l'on travaille, et non des problèmes de diffusion à résoudre. Cette tâche est donc effectuée une seule fois pour les trois exemples.

Remarque 7.1 *Le domaine étant homogène, les résultats (base et matrices locales) sont identiques d'une cellule à l'autre. Il n'est donc pas nécessaire de traiter toutes les cellules* $\hat{K}$*. Il est en effet plus simple et plus économique de travailler sur une unique cellule puis de reporter les résultats sur les autres cellules. Cette astuce de calcul est utilisée à nouveau dans l'Exemple 7.5, décrit à la section §7.2.2.*

Une fois les exemples résolus à l'échelle grossière, on dispose de leur solution grossière C_H. L'erreur normalisée maximale entre valeurs calculées et théoriques aux noeuds $I \in \mathscr{I}$ a été définie par la formule (5.7) :

$$e_\infty(C_H) = \max_{I \in \mathscr{I}} \frac{|C_H(I) - C(I)|}{1 + |C(I)|} \tag{7.2}$$

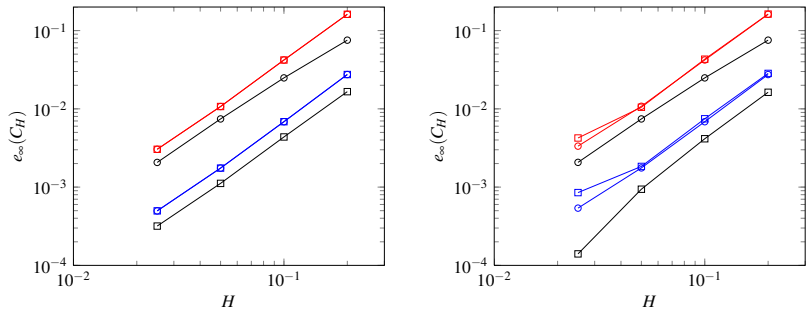


FIGURE 7.1 – Erreur $e_\infty(C_H)$ en fonction de H pour les Exemples 7.1.1 (—○—), 7.1.2 (—○—) et 7.1.3 (—○—) pour les méthodes d'Éléments Finis (—○—) et de Galerkin discontinue (—□—). Deux cas sont représentés : $\rho = 0$ (en haut) et $\rho = 0.5$ (en bas).

La Figure 7.1 présente l'évolution de l'erreur $e_\infty(C_H)$ en fonction de H lors de la résolution par la méthode d'Éléments Finis (cercle) et de Galerkin discontinue (carré) des exemples précédents : Exemple 7.1.1 (—○— et —□—), Exemple 7.1.2 (—○— et —□—) et Exemple 7.1.3 (—○— et —□—). Deux valeurs du sur-échantillonnage ρ sont représentées $\rho = 0$ (en haut) et $\rho = 0.5$ (en bas).

On retrouve ici des résultats similaires à ceux obtenus sur les exemples de qualification de l'implémentation du problème grossier, présentés à la Figure 5.12. En effet, que ce soit pour la méthode d'Éléments Finis ou celle de Galerkin discontinue, tous les exemples convergent en $\mathscr{O}(H^2)$ quand H tend vers 0.

Il est à noter que la méthode de Galerkin discontinue est plus sensible que la méthode d'Éléments Finis au choix du taux de sur-échantillonnage ρ, Comme on l'a présenté à la section §3.3, le calcul des termes matriciels des méthodes grossières dépend de la valeur de ρ. Si le sur-échantillonage est nul, les matrices sont calculées à partir des conditions aux limites des problèmes de cellules, qui sont explicitement connues. Dans le cas contraire, on interpole des valeurs aux faces de la frontière ∂K du macroélément K à partir des valeurs de la base locale sur la cellule $\hat{K}$ correspondante. La méthode de Galerkin discontinue faisant intervenir plusieurs matrices supplémentaires par rapport à la méthode d'Éléments Finis, notamment les matrices de saut $\mathbb{T}$ et de pénalité $\mathbb{Y}$, il est logique que l'influence du sur-échantillonage soit plus marqué.

Aux vues de ces résultats, on peut considérer que le couplage entre les parties locale et globale de la chaîne de calcul *SALOME-MPCube* est correctement implémenté.

7.1.2 Cas d'un milieu périodique.

7.1.2.1 Description du milieu.

On étudie maintenant les résultats des méthodes multi-échelles Q_1/*VFDiam* et GD/*VFDiam* sur un exemple théorique où le milieu D est périodique et oscillant.

Exemple 7.2 *On considère dans cet exemple le domaine* $\Omega =]0,1[^2$ *où l'on pose :*

$$D = \begin{bmatrix} \cos(2n\pi x)\cos(2n\pi y)+2 & 0 \\ 0 & \cos(2n\pi x)\cos(2n\pi y)+2 \end{bmatrix}$$

où n est un entier strictement positif. On choisit ici $n = 7$.

On résout le problème définit par :

$$\begin{aligned} f &= 2n\pi\left(\sin(2n\pi x)\cos(2n\pi y) - \cos(2n\pi x)\sin(2n\pi y)\right), \\ \Gamma_D &= \partial\Omega, \\ g_D &= x - y. \end{aligned}$$

La solution du problème est alors :

$$C = x - y.$$

Comme dans la section précédente, on discrétise le domaine Ω en un maillage $T_H(\Omega)$ composé de macroéléments carrés K de longueur H. Les cellules $\hat{K}$ sont ensuites construites puis discrétisées *via SALOME* pour obtenir les maillages $T_h(\hat{K})$ de pas $h = \alpha H$.

On traite ensuite chaque cellule $\hat{K}$ du domaine : résolution des problèmes locaux, détermination des bases locales $(\Phi_K^i)_{1\leq i\leq n}$ et enfin calcul des matrices locales. Cette étape achevée, on résout le problème par les méthodes d'Éléments Finis ou de Galerkin discontinue pour obtenir la solution grossière C_H. Le terme de pénalité μ de la méthode de Galerkin discontinue vaut ici 100.

À partir C_H et de la base $(\Phi^I)_{I\in\mathscr{I}}$, on reconstruit ensuite la solution fine $C_{H,h}$.

7.1.2.2 Base locale.

La Figure 7.2 présente la différence entre la fonction de base locale Φ_K^0 issue de la cellule $\hat{K} = (0,0)$ et la fonction $(1-x/H)(1-y/H)$, le polynôme de Lagrange correspondant à la méthode d'Éléments Finis Q_1 classique. On a choisit ici $H = 1/5$ et $\alpha = 1/20$. Trois valeurs de sur-échantillonnage sont illustrées : $\rho = 0$ (en

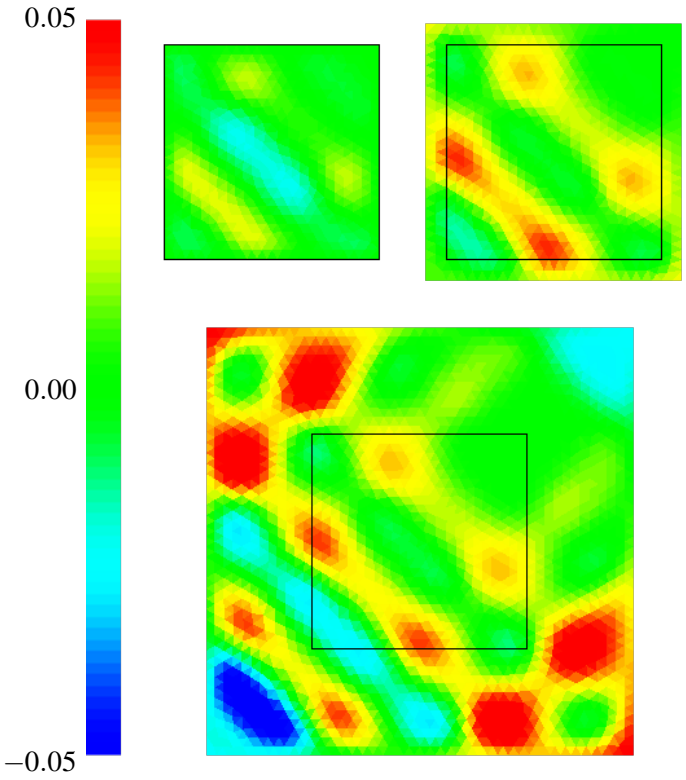


FIGURE 7.2 – Exemple 7.2 : différence entre la fonction de base locale Φ_K^0 issue de la cellule $\hat{K} = (0,0)$ et la fonction de référence $(1-x/H)(1-y/H)$. Pour $H = 1/5$, $\alpha = 1/20$, trois cas sont représentés : $\rho = 0$ (en haut à gauche), $\rho = 0.1$ (en haut à droite) et $\rho = 0.5$ (en bas). On a tracé en noir la frontière ∂K du macroélément.

haut à gauche), $\rho = 0.1$ (en haut à droite) et $\rho = 0.5$ (en bas). Les fonctions sont représentées sur la cellule $\hat{K}$ entière, le macroélément K étant délimité d'un trait noir.

L'influence du taux de sur-échantillonnage ρ est très visible sur cette illustration. Dans le cas où ρ est nul, la fonction de base Φ_K^0 ne présente que peu d'écart avec la fonction polynômiale classique. Elle n'est donc pas adaptée au milieu oscillant.

Au contraire, pour $\rho = 0.1$ et $\rho = 0.5$, on distingue nettement les oscillations de la fonction Φ_K^0. Comme on l'avait escompté, la technique de sur-échantillonnage remplit donc son rôle, en permettant de s'affranchir des contraintes de linéarités imposées par les conditions aux limites des problèmes de cellule, le phénomène de *couches limites* décrit à la section §3.3.1.1.

7.1.2.3 Erreurs à l'échelle fine.

Afin d'estimer l'erreur entre la solution fine $C_{H,h}$ et la solution théorique C, on introduit l'erreur L^2 discrète suivante :

$$e_2(C_{H,h}) = \sqrt{\sum_{K\in T_H(\Omega)} \sum_{k\in T_h(K)} |k|(C_{H,h}(k) - C(x_k))^2}, \tag{7.3}$$

où x_k est le barycentre de l'élément k du maillage fin $T_h(K)$.

Dans tous ce qui suit, on note $C^{EF}_{H,h}$ et $C^{GD}_{H,h}$ les solutions fines obtenues respectivement par les méthodes multi-échelles Q_1/*VFDiam* et GD/*VFDiam*.

La Figure 7.3 présente les erreurs $e_2(C^{EF}_{H,h})$ (cercle) et $e_2(C^{GD}_{H,h})$ (carré) en fonction de H. Chaque figure présente les erreurs pour trois valeurs de α : $\alpha = 1/5$ (—○— et —□—) $\alpha = 1/10$ (····○···· et ····□····), et $\alpha = 1/20$ (- ◡ - et - ⊔ -). Trois valeurs de ρ sont ici représentées : $\rho = 0$ (en haut), $\rho = 0.1$ (au milieu) et $\rho = 0.5$ (en bas).

La Figure 7.4 présente les erreurs $e_2(C^{EF}_{H,h})$ (cercle) et $e_2(C^{GD}_{H,h})$ (carré) en fonction de H lorsqu'on impose $\alpha = 1/10$. Trois valeurs de ρ sont représentées : $\rho = 0$ (—○— et —□—), $\rho = 0.1$ (····○···· et ····□····) et $\rho = 0.5$ (- ◡ - et - ⊔ -)

Remarque 7.2 *Comme on l'a expliqué à la section §4.2.1, les courbes présentées ici ne sont pas des courbes de* convergence *des méthodes* Q_1/VFDiam *et GD/*VFDiam*, mais des courbes d'*évolution *de l'erreur en fonction d'un paramètre (h, H ou* $\alpha = h/H$*), les autres étant fixés.*

Il apparaît sur ces courbes que les méthodes Q_1/*VFDiam* et GD/*VFDiam* ont sensiblement le même comportement. Les deux erreurs $e_2(C^{EF}_{H,h})$ et $e_2(C^{GD}_{H,h})$ décroissent globalement en $\mathcal{O}(H^{\frac{3}{2}})$ quand H diminue. On peut toutefois signaler que la méthode Galerkin discontinue donne de meilleurs résultats dans le cas d'un découpage très grossier ($H = 1/5$).

Pour les deux méthodes, l'évolution de l'erreur avec H n'est que peu influencée par les différents choix de α et de ρ. Comme on l'a expliqué à la section §3.3.1.1, le rôle du sur-échantillonnage dans les méthodes multi-échelles est d'éviter que les conditions aux limites des problèmes de cellules imposent leur linéarité sur les frontières des macroéléments K. La solution du problème étudié ici étant linéaire, cette linéarité artificielle ne pose pas problème. La technique de sur-échantillonnage est donc sans effet, puisqu'il n'y a pas de couche limite à corriger.

On présente à la Figure 7.5 une visualisation de l'écart $|C - C_{H,h}|$ entre la solution théorique C et la solution fine recalculée $C_{H,h}$ sur l'ensemble du domaine Ω. Pour un taux α fixé ($\alpha = 1/20$), on a représenté les écarts à l'échelle logarithmique pour $H = 1/5$ (en bas) et $H = 1/20$ (en haut) et pour deux valeurs de ρ : $\rho = 0$ (première et troisième lignes en partant du bas) et $\rho = 0.5$ (seconde et quatrième lignes). Les

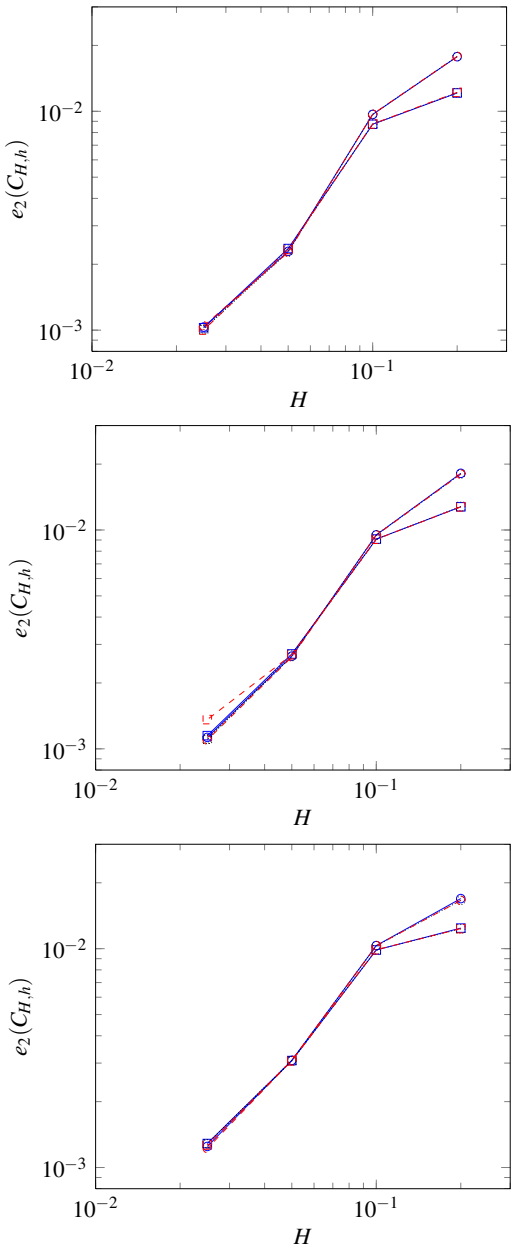


FIGURE 7.3 – Exemple 7.2 : erreurs $e_2(C^{EF}_{H,h})$ (cercle) et $e_2(C^{GD}_{H,h})$ (carré) en fonction de H. Chaque figure présente ici les erreurs pour trois valeurs de α : $\alpha = 1/5$ (—○— et —□—) $\alpha = 1/10$ (⋯○⋯ et ⋯□⋯), et $\alpha = 1/20$ (- ○ - et - □ -). Trois cas sont représentés : $\rho = 0$ (en haut), $\rho = 0.1$ (au milieu) et $\rho = 0.5$ (en bas).

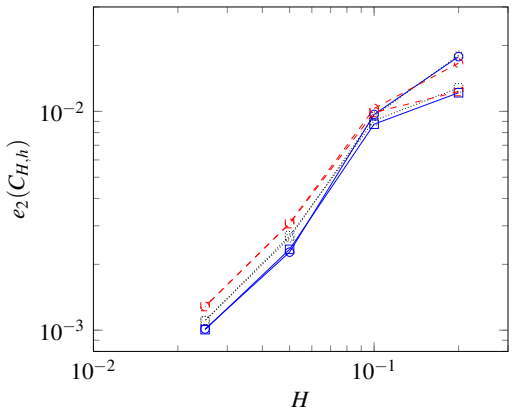


FIGURE 7.4 – Exemple 7.2 : erreurs $e_2(C^{EF}_{H,h})$ (cercle) et $e_2(C^{GD}_{H,h})$ (carré) en fonction de H, à $\alpha = 1/10$ fixé. On présente ici les erreurs pour trois valeurs de ρ : $\rho = 0$ (—○— et —□—), $\rho = 0.1$ (⋯○⋯ et ⋯□⋯) et $\rho = 0.5$ (- -○- - et - -□- -).

deux méthodes grossières sont illustrées : Éléments Finis (à gauche) et Galerkin discontinue (à droite).

Dans le cas $H = 1/5$, on aperçoit les avantages qualitatifs de la méthode de Galerkin discontinue sur la méthode Éléments Finis. Le maximum de l'erreur est plus faible, et l'erreur en elle-même est mieux répartie. En particulier, dans le cas où le sur-échantillonnage ρ vaut 0.5, les sauts de discontinuités sont plus amortis par la méthode de Galerkin discontinue. Ce lissage des discontinuités se retrouve également quand H vaut $1/20$, même si son intérêt est moindre car les sauts de discontinuités y sont naturellement plus faible qu'au cas précédent.

7.1.3 Réponse de la chaîne de calcul à un grand nombre de macroéléments.

Exemple 7.3 *On considère le problème suivant :*

$$\begin{cases} -\Delta C &= 0 & \text{dans } \Omega =]0,1[^3 \\ C &= xyz & \text{sur } \Gamma = \partial\Omega. \end{cases} \tag{7.4}$$

La solution du problème est alors :

$$C = xyz$$

On divise le domaine en 50 *macroéléments dans chaque direction et on utilise un sur-échantillonnage de 10%. On a donc* $\mathscr{D} = (50,50,50)$ *et* $\rho = 0.1$.

L'intérêt de l'Exemple 7.3 ne réside pas dans la résolution proprement dite du problème. La solution étant linéaire, les méthodes multi-échelles permettent de déterminer la solution à la précision machine près.

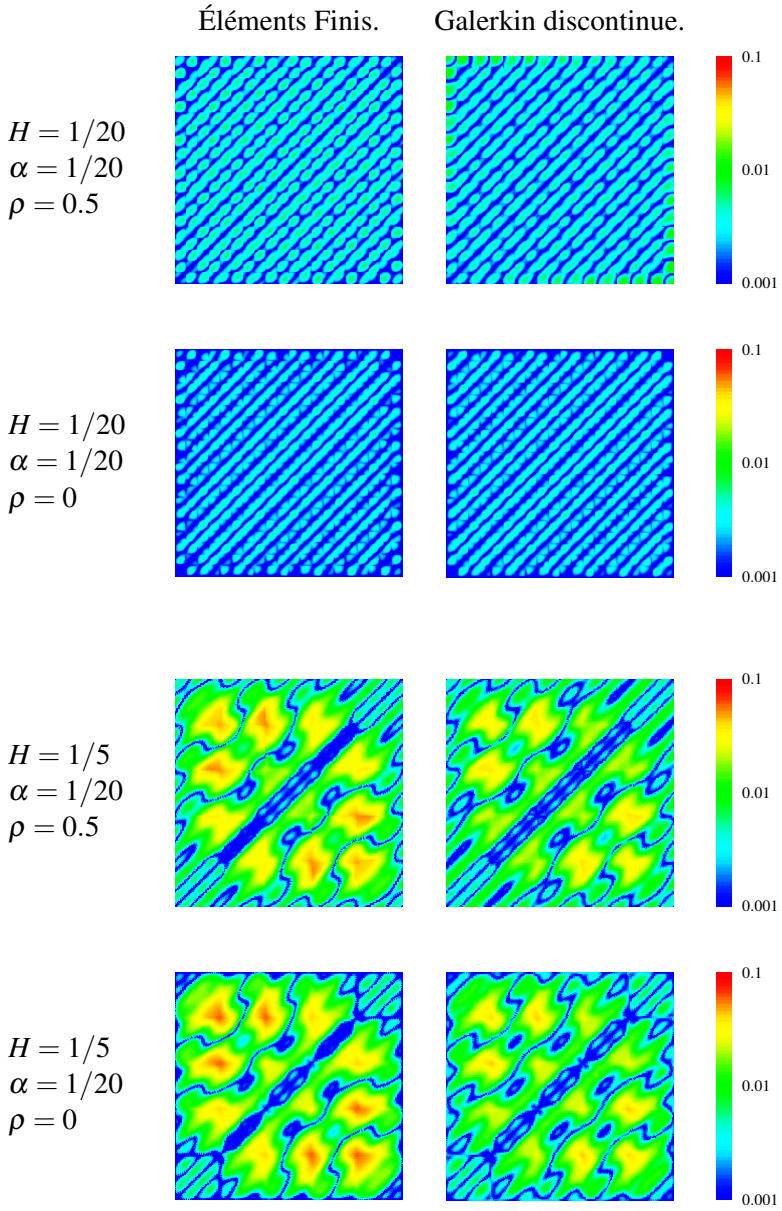


FIGURE 7.5 – Exemple 7.2 : écart entre solution théorique C et solution fine recalculée $C_{H,h}$ pour les méthodes Éléments Finis (à gauche) et Galerkin discontinue (à droite). On a fixé $\alpha = 1/20$ et représenté les écarts à l'échelle logarithmique pour $H = 1/5$ (en bas) et $H = 1/20$ (en haut) et pour deux valeurs de ρ : $\rho = 0$ et $\rho = 0.5$.

Le but de cet exemple est de tester la réponse de la chaine de calcul *SALOME-MPCube* à une division importante du milieu. Les méthodes multi-échelles sont en effet conçues pour résoudre des problèmes de grandes et très grandes tailles, et il est donc important que l'architecture informatique implémentant ces méthodes soit robuste.

Avec $\mathscr{D} = (50, 50, 50)$, on construit un maillage grossier $T_H(\Omega)$ de 125000 éléments. Il est donc nécessaire de construire 125000 maillages fins $T_h(\hat{K})$ de cellules, puis de résoudre un million de problèmes de cellules (8 par cellule).

7.1.3.1 Stockage et manipulation de fichiers.

Un problème concernant la gestion des capacités de stockage a été révélé par cette simulation. Elle a mis en évidence la nécessité d'utiliser une arborescence de fichiers à plusieurs niveaux afin de sauvegarder les données manipulées. En effet, le nombre de fichiers et de répertoires que peut contenir un répertoire est limité, les chiffres exacts dépendant du système de fichiers choisi (ext3, ext4, Reiser4, FAT32, exFAT, etc.).

Ces limites sont en deça des besoins de l'Exemple 7.3, même pour les plus récents des systèmes de fichiers. Par exemple, au sein d'un système de fichier ext4 le nombre maximal de sous-répertoires d'un même répertoire est 64000. Le nombre maximal de fichiers par répertoire au sein d'un système exFAT est de 2796202, ce qui permet dans cet exemple un maximum de 22 fichiers par cellule.

Afin de s'affranchir de ces limites, les cellules se voient attribuer une arborescence unique en fonction de leur numéro. Pour la cellule $\hat{K}$ numérotée (p,q,r), l'arborescence choisie est *./wXp/wYq/wZr*. Cette arborescence est ajoutée aux répertoires de travail ; utilisé lors de la résolution des problèmes de cellules ; et de stockage, où l'on enregistre la base et les matrices locales. Ainsi, si on nomme *./Work* et *./Data* les répertoires de travail et de stockage, le travail sur la cellule $(2,1,5)$ utilise le répertoire *./Work/wX2/wY1/wZ5*. Les résultats sont ensuite stockés dans le répertoire *./Data/wX2/wY1/wZ5*.

7.1.3.2 Génération des maillages.

On construit les maillages $T_h(\hat{K})$ des cellules $\hat{K}$ issues des macroéléments K de $T_H(\Omega)$. Les maillages sont générés les uns après les autres au sein de *SALOME*. Ils ne sont pas coïncidents (cf. §6.3) et se composent d'environ 4000 tétrahèdres. Ce nombre peut paraître peu élevé, mais on rappelle que le but de cet exemple n'est pas d'obtenir une simulation précise, mais d'étudier la réponse de la chaîne de calcul à une division importante du milieu.

On présente à la Figure 7.6 le temps nécessaire à la génération des maillages. On utilise pour cela un noeud du serveur de calcul *pax*, un biprocesseur *AMD Opteron 252* (cadence 2.6Ghz, mémoire vive 16Go). Seuls les 8000 premiers maillages ont été représentés.

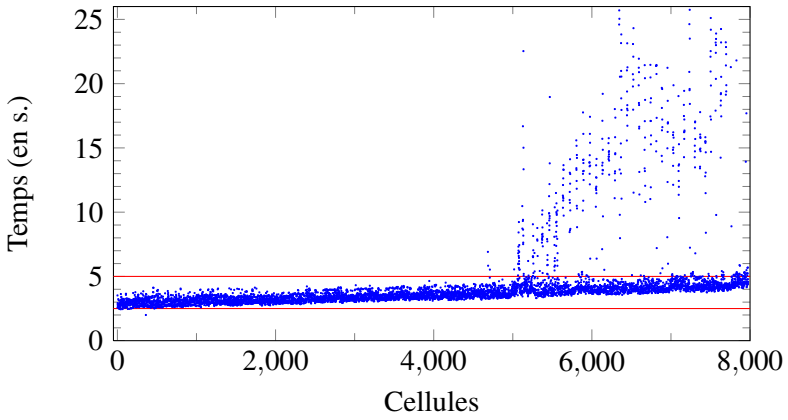


FIGURE 7.6 – Exemple 7.3 : temps de génération de huit mille maillages fins $T_h(\hat{K})$ en seconde. Le temps moyen augmente au fur et à mesure du nombre de maillages à cause de fuites mémoires dans *SALOME*. Quand la mémoire est saturée, le temps de génération peut être multiplié par 5 par les opérations de *swap* et de nettoyage mémoire.

Le temps de génération augmente au fur et à mesure du nombre de maillages, passant de $2.5s$ à $5s$, pour une moyenne de $4.39s$. Cette augmentation est causée par des fuites mémoires au sein de *SALOME* déclenchées par les routines Python de génération de maillages. Un travail conséquent a été fourni pour rendre ces routines, décrites en détail au chapitre §6, neutres du point de vue de la mémoire. Cependant, la plate-forme *SALOME* a des difficultés de gestion de mémoire, et les ressources allouées à un objet ne sont pas nécessairement libérées quand l'utilisateur demande la destruction de l'objet.

Quand la mémoire disponible commence à manquer, la génération du maillage est interrompue, le temps que le serveur de calcul utilisent ses fonctionnalités de nettoyage mémoire, ou procède au *swap* de données. Le temps de génération peut ainsi être multiplié par 5 ou plus. Plus la mémoire est saturée, plus ces générations «interrompues» sont fréquentes, jusqu'à devenir la règle plutôt que l'exception. À terme, le travail s'interrompt faute de mémoire.

Afin de pallier ce problème, il suffit de quitter périodiquement l'instance *SALOME* en cours pour en lancer une nouvelle. Le travail de génération peut alors reprendre. Les routines Python de génération des maillages ont été adaptées en ce sens : elles peuvent travailler sur un nombre limité de cellules plutôt que sur l'ensemble des cellules. Il revient cependant à l'utilisateur de décider de la valeur de ce nombre en fonction des ressources informatiques disponibles.

Cette solution mise en place, le gain de temps est important. Si on génère les maillages par groupe de 1000, le temps moyen de construction est de $3.23s$, contre $4.39s$ précédemment.

7.1.3.3 Résolution des problèmes locaux.

Pour traiter les cellules, on utilise un cluster de calcul, nommé *awa*, composé d'un noeud maître quadiprocesseur *AMD Opteron 852* (cadence 2.6Ghz) et de six noeuds esclaves, chaque noeud étant un biprocesseur *AMD Opteron 250* (cadence 2.4Ghz, mémoire vive 4Go).

On traite les cellules cinq par cinq, chaque cellule se voyant attribuer un noeud, et donc deux cœurs, de calcul. Comme expliqué à la section §5.1.3, une instance *MPCube* est lancée pour chaque quintuplet de cellules. Sur chaque cellule, on résout les huit problèmes de cellules, puis on calcule les matrices locales correspondantes.

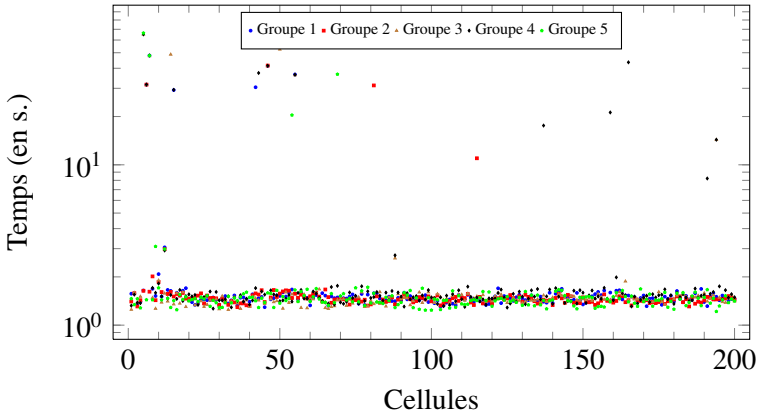


FIGURE 7.7 – Exemple 7.3 : temps de résolution des problèmes locaux et de calcul des matrices locales, en seconde. La majorité des cellules est traitée entre $1s$ et $2s$.

On présente à la Figure 7.7 la durée en seconde de ces opérations. 200 traitements, soit 1000 cellules au total réparties en 5 groupes, sont représentés. Le temps de travail oscille dans la très grande majorité des cas entre $1s$ et $2s$. On peut en conclure qu'il n'y a pas de pertes mémoires lors de ces opérations.

Certaines cellules sont traitées en un temps anormalement élévé, de l'ordre de quelques dizaines de secondes. Dans la majorité des cas, ces écarts se retrouvent sur plusieurs cellule d'un même groupe. Ils ne dépendent donc pas des cellules traitées proprement dites, mais du quintuplet en entier, et donc de l'instance *MPCube* correspondante. On attribue donc ces écarts à une utilisation des noeuds de calcul par des programmes tierces, extérieurs à la chaîne de calcul multi-échelle.

7.1.3.4 Mise en forme des résultats.

Telle qu'on l'a décrite à la section §5.1, la chaîne de calcul *SALOME-MPCube* met en forme les résultats locaux (base locale $(\Phi_K^i)_{1\leq i\leq n}$, flux) une fois terminé le

travail sur les cellules : résolution des problèmes de cellules (§5.1.3) puis construction de la base locale et calcul des matrices locales (§5.1.4). La mise en forme des données, détaillée à la section §5.1.5, permet de concentrer les résultats sur un seul fichier, tout en supprimant les fichiers temporaires crées plus tôt dans la chaîne de calcul. On ne peut attribuer qu'un seul cœur de calcul par cellule à cette tâche, mais plusieurs cellules peuvent être formatées en même temps (parallélisme *extra*-cellulaire).

Afin de limiter le nombre de fichiers temporaires et les besoins en matière de stockage, la chaîne de calcul a été modifiée afin d'alterner les étapes de travail et de mise en forme. Ainsi, si on attribue p cœurs de calcul à la résolution des problèmes locaux, une étape de mise en forme est effectuée après p étapes de travail. Dans l'Exemple 7.3, on traite donc 2 jeux de 5 cellules avant de mettre en forme les résultats des 10 cellules correspondantes.

7.2 Applications aux matériaux cimentaires.

7.2.1 Application à un échantillon de mortier.

7.2.1.1 Description du milieu.

On s'intéresse dans cette section aux mortiers. Il s'agit de mélanges de poudre de ciment, de sable et d'eau que l'on a laissé reposé et séché. La présence ou non de grains de sables au sein de la pâte de ciment a une forte influence sur la diffusion des ions au sein du matériau. En effet, lors de la fabrication du mortier, on observe la formation d'une *auréole de transition*, de très haute diffusivité, autour des grains de sables [52, 176]. Alors que les grains de sables sont nécessairement séparés les uns des autres, les auréoles de transition correspondantes peuvent fusionner les unes les autres. Au delà d'une certaine concentration de sable, de véritables chemins de haute diffusivité apparaissent à travers le milieu, un phénomène appelé *percolation* [168].

Les mortiers et les matériaux cimentaires en général sont détaillés à la section §3.6. En particulier la Figure 3.6 illustre le phénomène de percolation.

Les résultats présentés dans cette section ont été partiellement rédigés dans un article [14], reproduit à l'Annexe §C.

Exemple 7.4 *On considère un domaine cubique* $\Omega = [0,L]^3$*, avec* $L = 5$*, représentant un échantillon de* 125mm^3 *de mortier. Il est composé d'une matrice, modélisant la pâte de ciment, de diffusivité adimensionnée* $D_m = 5$*. Cette matrice intègre des inclusions sphériques, les grains de sable, de diffusivité* $D_s = 1$*. Chaque sphère est enveloppée d'une couche de* $30\mu\text{m}$ *d'épaisseur, de diffusivité* $D_t = 15$*, modélisant l'auréole de transition.*

Les diffusivités ont été mesurées par porosimétrie au mercure sur un échantillon de mortier avant d'être adimensionnées [52]. Le sable occupe 35% du volume total

du domaine, soit environ 2600 inclusions. La répartition des tailles des grains de sable est issues de mesures industrielles par granulométrie [89]. Elle est détaillée à la Figure 7.8.

Les fonctionnalités *SALOME* développées au cours de ces travaux de thèse travaillent sur des jeux d'inclusions disjointes les unes des autres. Pour ces raisons techniques, on suppose donc que les auréoles de transition sont distinctes les unes des autres, ainsi donc que les grains de sable correspondants, bien que ce soit physiquement peu probable étant donnée le volume relativement important occupé par le sable.

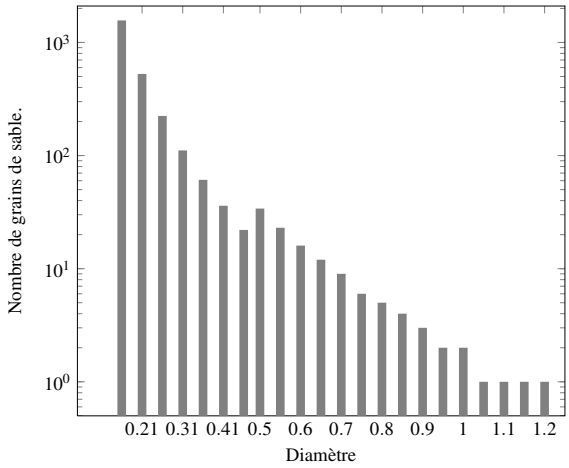


FIGURE 7.8 – Exemple 7.4 : distribution de la taille des grains de sable dans le ciment, calculée à partir des courbes granulométriques du sable. On a exclu les plus grandes tailles, de 1.2mm à 4mm, car le domaine de travail est long de 5mm seulement.

7.2.1.2 Résolutions de problèmes.

Le domaine est divisé en 5 macroéléments dans chaque direction de l'espace, $\mathscr{D} = (5,5,5)$ pour un total de 125 macroéléments. On n'a pas utilisé la technique de sur-échantillonnage : ρ est nul.

Chaque cellule est discrétisée en utilisant les outils de maillage BLSURF et GHS3D *via* la plate-forme *SALOME*, comme on l'a présenté au chapitre §6. Les maillages obtenus comportent entre 1×10^5 et 5×10^5 volumes tétrahédriques. Cet écart est du aux spécificités locales de la géométrie de la cellule. Par exemple, la présence de points de tangences entre les inclusions sphériques et les frontières de la cellule augmente énormément le nombre de tétrahèdres nécessaires à une discrétisation appropriée de la cellule. On discute ce point plus en détails à la section §6.2.4.

Sur l'ensemble du domaine Ω, la discrétisation totalise approximativement 2×10^7 éléments. La Figure 7.9 présente la maillage d'une cellule, comportant près de

2×10^5 éléments.

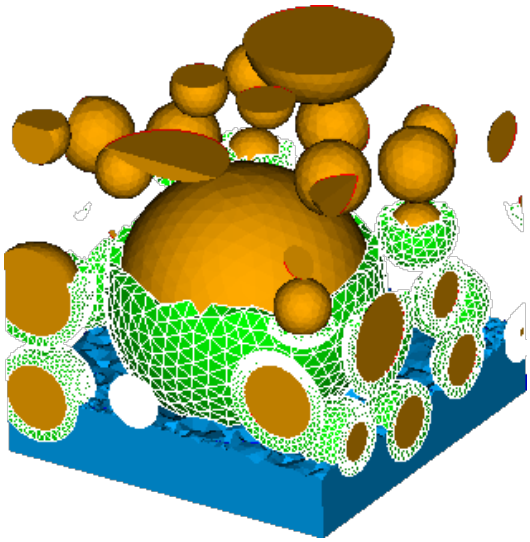


FIGURE 7.9 – Exemple 7.4 : maillage d'une cellule du domaine. La pâte de ciment (en bas) n'est que partiellement représentée afin de montrer les grains de sable sphériques et leurs auréoles de transition (en surbrillance, invisibles dans la partie supérieure de la cellule).

Exemple 7.4.1 *On note $\Gamma_{x,m}$ et $\Gamma_{x,p}$ les faces du domaine Ω pour lesquelles on a $\{x = 0\}$ et $\{x = L\}$ respectivement. On définit de même les faces $\Gamma_{y,m}$, $\Gamma_{y,p}$, $\Gamma_{z,m}$, $\Gamma_{z,p}$.*

Sur l'échantillon de mortier, on résout le problème de diffusion suivant :

$$\begin{cases} -\nabla(D\nabla C) &= 0 \quad \textit{dans } \Omega, \\ C &= 1 \quad \textit{sur } \Gamma_{x,m}, \\ C &= 0 \quad \textit{sur } \Gamma_{x,p}, \\ D\nabla C \cdot \mathbf{n} &= 0 \quad \textit{sur les autres faces,} \end{cases} \tag{7.5}$$

où $\mathbf{n}$ désigne le vecteur normal extérieur à Ω.

Avec les même notations, on définit deux exemples similaires à l'Exemple 7.4.1, la différence tenant dans les faces du domaine Ω où l'on impose les conditions limites de Dirichlet.

Exemple 7.4.2

$$\begin{cases} -\nabla(D\nabla C) &= 0 \quad \textit{dans } \Omega, \\ C &= 1 \quad \textit{sur } \Gamma_{y,m}, \\ C &= 0 \quad \textit{sur } \Gamma_{y,p}, \\ D\nabla C \cdot \mathbf{n} &= 0 \quad \textit{sur les autres faces.} \end{cases} \tag{7.6}$$

Exemple 7.4.3

$$\left\{\begin{array}{rcll} -\nabla(D\nabla C) & = & 0 & \textit{dans } \Omega, \\ C & = & 1 & \textit{sur } \Gamma_{z,m}, \\ C & = & 0 & \textit{sur } \Gamma_{z,p}, \\ D\nabla C\cdot\mathbf{n} & = & 0 & \textit{sur les autres faces.} \end{array}\right. \tag{7.7}$$

On résout les problèmes des Exemples 7.4.1, 7.4.2 et 7.4.3 par la méthode multi-échelle Q_1/*VFDiam*, puis on recalcule les solutions fines $C^x_{H,h}$, $C^y_{H,h}$ et $C^z_{H,h}$ correspondantes.

On présente à la Figure 7.10 la solution $C^x_{H,h}$ et le gradient longitudinal $\nabla_x C^x_{H,h}$. On représente des coupes des coupes des solutions pour les valeurs de x allant de 1 à 3, les coordonnées (y,z) étant fixées. Les sauts de concentration, et surtout de gradient, identifient le passage de la solution à travers les auréoles de transition.

7.2.1.3 Calcul de la diffusivité équivalente.

Les problèmes que l'on résout ici reproduisent les mesures expérimentales de diffusivité équivalente [48, 49, 63, 73]. L'Exemple 7.4.1 permet ainsi de calculer D^*_x, une *diffusivité équivalente du milieu* D^* selon la direction x, à partir des flux entrant et sortant :

$$D^*_{x,m} = -\frac{1}{L}\int_{\Gamma_{x,m}} D\nabla C\cdot\mathbf{n}, \tag{7.8}$$

$$D^*_{x,p} = \frac{1}{L}\int_{\Gamma_{x,p}} D\nabla C\cdot\mathbf{n}. \tag{7.9}$$

Compte tenu des propriétés de la solution C de l'Exemple 7.4.1, on a :

$$D^*_{x,m} = D^*_{x,p}$$

La notion de diffusivité équivalente est liée à la définition d'un *Volume Élémentaire Représentatif* (VER). Il s'agit d'un échantillon type du matériau, dont on suppose qu'il renferme toutes les structures, toutes les informations nécessaires pour prédire correctement le comportement macroscopique du matériau. À l'échelle macroscopique, un volume quelconque du matériau pourra donc être remplacé par un assemblage de ses VER sans affecter les résultats des simulations.

La diffusivité équivalente D^* doit être cohérente avec la définition du VER du milieu et avec le comportement attendu à l'échelle supérieure. On considère généralement dans la littérature que le mortier se comporte comme un matériau homogène isotrope [52]. Les valeurs de D^* calculées à partir du VER doivent donc l'être aussi.

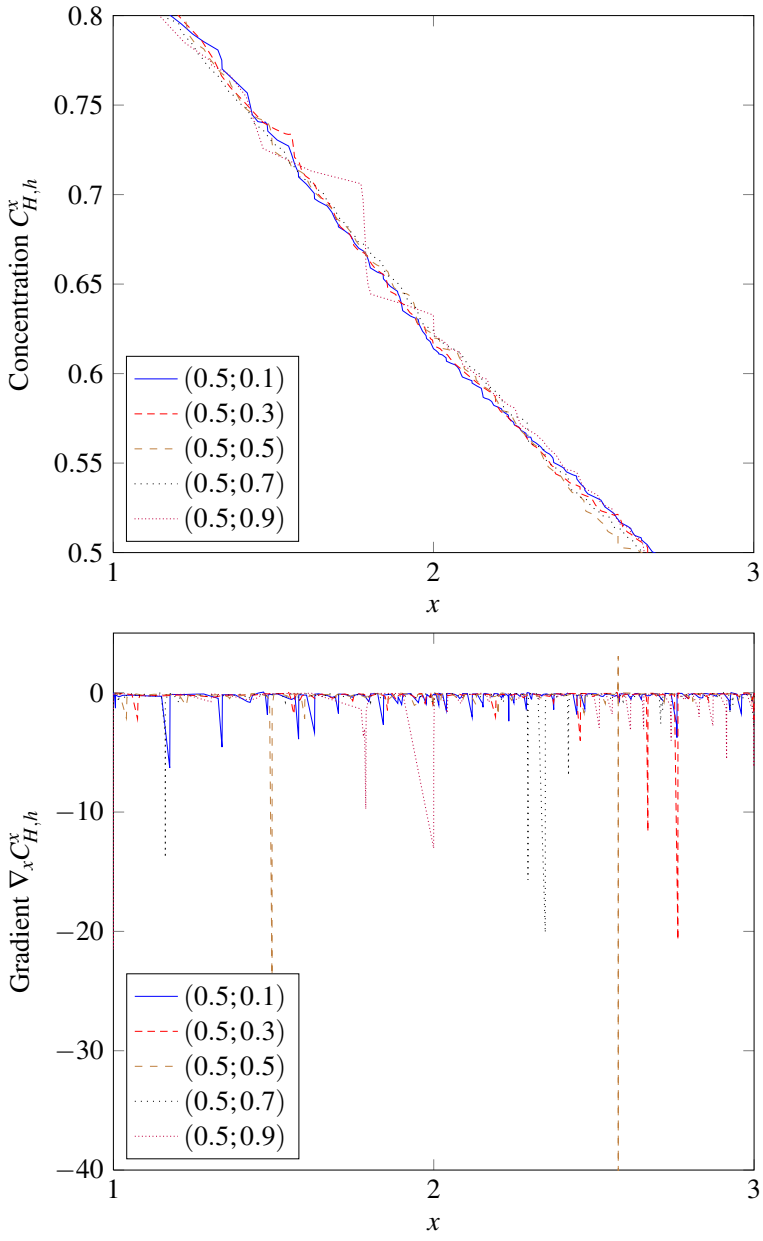


FIGURE 7.10 – Exemple 7.4.1 : concentration $C^x_{H,h}$ et gradient longitudinal $\nabla_x C^x_{H,h}$ à travers le domaine Ω. Les coupes s'étendent de $x = 1$ à 3, les coordonnées $(y;z)$ étant constantes. On reconnaît facilement les *auréoles de transition* par les sauts qu'elles provoquent sur les profils de concentration.

	$D^*_{.,m}$	$D^*_{.,p}$
$D^*_{x,.}$	8.41	10.43
$D^*_{y,.}$	8.70	8.48
$D^*_{z,.}$	9.52	9.07

TABLE 7.1 – Exemple 7.4 : diffusivités équivalentes D^* calculées selon les deux directions des trois axes de l'espace. Les diffusivités sont différentes les unes des autres, l'échantillon choisi du milieu n'est donc pas représentatif.

À partir des solutions fines $C^x_{H,h}$, $C^y_{H,h}$ et $C^z_{H,h}$, on calcule six diffusivités équivalentes, que l'on présente au Tableau 7.1.

Selon les directions y et z, l'écart entre les diffusivités $D^*_{.,m}$ et $D^*_{.,p}$ est faible (moins de 5%). L'écart est en revanche très net selon la direction x (25%). De plus le milieu homogénéisé n'est pas isotrope, les diffusivités équivalentes n'étant pas identiques selon chaque direction. L'échantillon ici considéré n'est donc pas représentatif du mortier.

Ces résultats peuvent être imputés d'une part à la petite taille physique de l'échantillon par rapport à celles des inclusions et au découpage grossier du domaine généré d'autre part.

En effet, on considère ici un échantillon de 5mm de coté, alors que certains grains de sables font jusqu'à 4mm de long. Par conséquent, le plus grand grain de sable présent dans cet échantillon a un diamètre de 1.2mm, et seuls 6 grains de sable sur 2600 ont un diamètre supérieur ou égal à 1mm. Compte tenu de leurs volumes respectifs, la structure des macroéléments change donc du tout au tout selon qu'ils possèdent ou non l'un de ces «grands» grains de sable. Cette variété des macroéléments se transmet naturellement aux fonctions de bases locales $(\Phi^i_K)_{1\leq i\leq n}$, aux matrices locales et aux flux sortants.

Cependant, on a ici découpé le domaine en 5 macroélément selon chaque direction. Compte tenu des conditions limites de Dirichlet, le nombre réel de dégrés de liberté du problème grossier est très faible, trop pour que les variations des matrices locales se retrouvent dans la solution grossière. La solution grossière C_H décroît donc uniformément, sans tenir compte des différences entre les macroéléments.

Lorsqu'on reconstruit la solution fine $C_{H,h}$ on pondère donc les fonctions oscillantes $(\Phi^I)_{I\in\mathscr{I}}$ par des coefficients qui ne tiennent pas compte de leurs spécificités. Toute différence sur la structure des macroéléments se répercute alors sur la solution fine, et sur son flux, sans aucune contrepartie. En particulier, puisqu'un des grands grains de sable, et son auréole de transition, occupe une importante portion de la face $\Gamma_{x,p}$, le flux de $C_{H,h}$ sur cette face du domaine est anormalement élevé, ce qui explique l'écart de valeur de $D^*_{x,p}$. Cette différence ne se retrouve pas sur les autres faces de Ω.

En conclusion, on a montré par le biais de cet exemple l'utilité de la méthode multi-échelle Q_1/*VFDiam*, en permettant de réaliser facilement plusieurs simulations sur un domaine nécessitant une discrétisation fine (environ 2×10^7 éléments).

L'échantillon modélisé n'est cependant pas représentatif du mortier. Pour cela, il conviendrait de travailler sur un domaine plus large, de l'ordre de 20mm ou 50mm de côté, afin de prendre en compte toute l'étendue des tailles de grains de sable.

7.2.2 Application à un échantillon périodique de pâte de ciment.

7.2.2.1 Description du milieu.

Cet exemple se concentre sur un échantillon de pâte de ciment qui sera reproduit périodiquement pour former le domaine Ω. Ici, le macroélément K n'est pas défini à partir d'un découpage du domaine Ω, mais c'est l'échantillon Ω qui est construit par assemblage périodique du macroélément K, que l'on peut donc considérer comme un VER. En ce sens, on travaille donc *a contrario* de l'ordre habituel.

On appelle *pâte de ciment* une poudre de ciment qui a été hydratée, brassée puis séchée. Pendant ces opérations, de nombreuses transformations chimiques ont lieu, ce qui entraîne l'apparition de nouvelles espèces chimiques, ainsi que le réarrangement de composés en de nouvelles structures [115, 150]. On peut trouver plus de détails sur la pâte de ciment et ses principaux composés à la section §3.6.1.

Exemple 7.5 *On considère un macroélément cubique $K = [0,L]^3$, avec $L = 50\mu$m, représentant un échantillon de $125\mu\text{m}^3$ de pâte cimentaire. Il est composé d'une matrice, modélisant la porosité capillaire, de diffusivité $D_0 = 2.24 \times 10^{-9}\text{m}^2.\text{s}^{-1}$.*

Cette matrice intègre des inclusions sphériques modélisant les composés principaux de la pâte de ciment. Les phases non diffusives (résidus anhydres, portlandite, ettringite et hydrogrenats) sont fixées à une valeur basse (diffusivité $D_a = 1 \times 10^{-20}\text{m}^2.\text{s}^{-1}$). Les silicates de calcium hydratés *ou $C-S-H$ sont répartis en deux densités différentes :* Low Density *$C-S-H$ (diffusivité $D_l = 9 \times 10^{-12}\text{m}^2.\text{s}^{-1}$) et* High Density *$C-S-H$ (diffusivité $D_h = 1 \times 10^{-12}\text{m}^2.\text{s}^{-1}$).*

La majorité des inclusions sont de petites tailles (rayon r inférieur à 5μm), mais il existe aussi des inclusions plus grandes ($r > 6\mu$m) possédant une structure multi-couche : un coeur de portlandite (ou d'aluminates, d'anhydre) occupant 40% du rayon, une coquille de HD *$C-S-H$ (de $0.4r$ à $0.9r$) et une coquille de* LD *$C-S-H$ (de $0.9r$ à r).*

La minéralogie de la pâte de ciment et la diffusivité des différentes phases correspondent à une pâte de ciment *CEMI* [42, 43, 45]. On présente à la Figure 7.11 (cas $\rho = 0$) la distribution de la taille des inclusions dans le macroélément K. Les barres présentent la répartition des inclusions sur les différentes phases considérées.

Ces inclusions sont placées aléatoirement au sein du macroélément K par le module COMBS, décrit à la section §4.3.1.

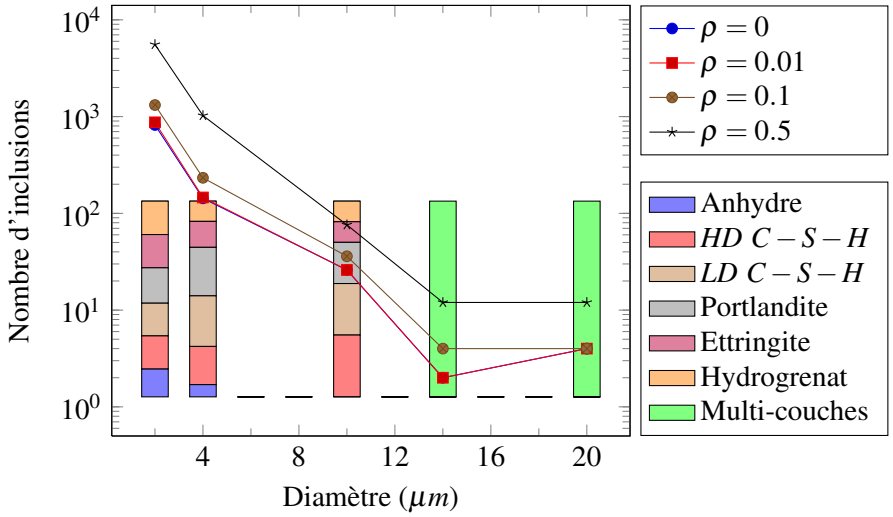


FIGURE 7.11 – Exemple 7.5 : distribution de la taille des inclusions dans la cellule périodique en fonction de ρ. Les barres présentent la répartition des inclusions sur les différentes phases considérées.

7.2.2.2 Travail sur le macroélément K.

Les différentes étapes de la chaîne de calcul *SALOME-MPCube* sont appliquées au macroélément K.

Soit ρ un taux de sur-échantillonage donné, on construit et on discrétise la cellule $\hat{K}$ correspondante par le biais de la plate-forme *SALOME*. Les méthodes utilisées pour obtenir $T_h(\hat{K})$, la discrétisation de la cellule $\hat{K}$, ont été décrites en détails au chapitre §6, et plus précisément à la section §6.3. De légères modifications ont cependant été effectuées afin d'assurer que les faces opposées du macroélément K soient discrétisées de manière identique.

Le Tableau 7.2 présente, pour quatre valeurs de ρ, le nombre d'inclusions présentes dans la cellule $\hat{K}$ et le nombre de tétrahèdres des maillages $T_h(\hat{K})$ correspondants. Une visualisation du maillage du macroélément K dans le cas $\rho = 0$ se trouve à la Figure 7.12.

ρ	Nombre d'inclusions	Nombre de tétrahèdres
0	594	1 110 928
0.01	594	1 217 065
0.1	1115	1 889 902
0.5	5486	8 877 141

TABLE 7.2 – Exemple 7.5 : nombre d'inclusions de la cellule $\hat{K}$ et nombre de tétrahèdres du maillage $T_h(\hat{K})$ pour plusieurs valeurs de ρ.

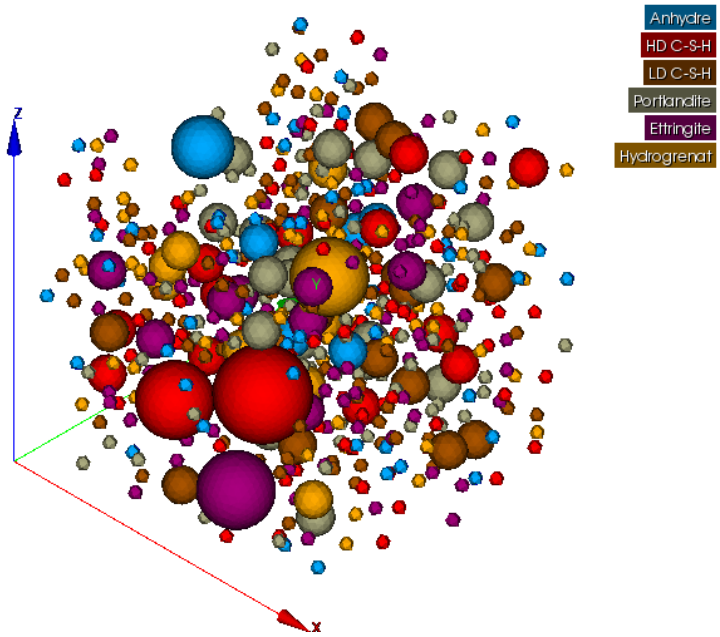


FIGURE 7.12 – Exemple 7.5 : visualisation *via SALOME* du maillage du macro-élément K, dans le cas $\rho = 0$. Seul le maillage des inclusions est ici représenté, la matrice ayant été rendue invisible. Chaque couleur correspond à une phase différente.

Remarque 7.3 *Pour des raisons techniques, on doit ignorer les inclusions qui touchent l'une ou l'autre des faces du macroélément K. Les chiffres présentés au Tableau 7.2 sont donc inférieurs à ceux présentés à la Figure 7.11, qui correspondent aux nombres d'inclusions générées par le module* COMBS. *On revient plus en détails sur ce problème à la Remarque 7.4.*

Une fois la cellule $\hat{K}$ discrétisée, on résout les problèmes de cellules par le biais de *MPCube*. On calcule ensuite la base locale $(\Phi_K^i)_{1\leq i\leq n}$ et les matrices locales associées, ce qui achève le travail sur la cellule $\hat{K}$.

7.2.2.3 Résolution de problèmes grossiers.

Soit p un entier strictement positif, on note Ω^p le domaine cubique $[0, pL]^3$ composé de l'assemblage de p^3 exemplaires du macroélément K décrit ci-dessus.

Compte tenu de sa périodicité, la résolution d'un problème sur le milieu Ω^p par une méthode multi-échelle a un coût très faible. En effet, les résultats des calculs locaux (base et matrices locales) sont ici identiques d'une cellule à l'autre. Une fois effectués ces calculs sur l'unique cellule $\hat{K}$ de référence, il suffit donc de les reporter sur les autres cellules pour pouvoir assembler le problème grossier.

On résout sur Ω^p les trois problèmes d'homogénéisation précédemment décrit à la section §7.2.1.2 sous les noms des Exemples 7.4.1, 7.4.2 et 7.4.3. Ce problème est définit, pour la direction x, par :

Exemple 7.5.1

$$\left\{\begin{array}{rcll} -\nabla(D\nabla C) & = & 0 & \textit{dans } \Omega^p, \\ C & = & 1 & \textit{sur } \Gamma^p_{x,m}, \\ C & = & 0 & \textit{sur } \Gamma^p_{x,p}, \\ D\nabla C\cdot\mathbf{n} & = & 0 & \textit{sur les autres faces,} \end{array}\right. \tag{7.10}$$

où $\mathbf{n}$ *désigne le vecteur normal extérieur à* Ω^p *et où on a noté* $\Gamma^p_{x,m}$ *et* $\Gamma^p_{x,p}$ *les faces du domaine* Ω^p *pour lesquelles on a* $\{x=0\}$ *et* $\{x=pL\}$ *respectivement. On définit de même les faces* $\Gamma^p_{y,m}$, $\Gamma^p_{y,p}$, $\Gamma^p_{z,m}$, $\Gamma^p_{z,p}$.

Par le biais des équations (7.8) et (7.9), on construit six diffusivités équivalentes à partir des solutions des problèmes d'homogénéisation. On présente au Tableau 7.3 l'évolution de ces valeurs en fonction de p pour les deux méthodes d'Éléments Finis et de Galerkin discontinue. Deux valeurs de ρ sont représentées : $\rho = 0$ et $\rho = 0.1$.

Étant donné la périodicité des milieux Ω^p, leurs diffusivités équivalentes ne doivent pas évoluer avec p, mais correspondre aux diffusivités équivalentes du macroélément K de référence. De plus, on considère dans la littérature que la pâte cimentaire se comporte comme un matériau homogène isotrope [42, 43]. Les valeurs de D^* calculées ici doivent donc l'être aussi.

Dans le cas où on n'utilise pas de sur-échantillonnage ($\rho = 0$), les diffusivités calculées varient très peu avec p, que ce soit pour la méthode Éléments Finis ou celle de Galerkin discontinue. En outre, elles sont très proches les unes des autres. En effet, que ce soit selon les directions ($D^*_{.,m}$ et $D^*_{.,p}$), selon les axes ($D^*_{x,.}$, $D^*_{y,.}$ et $D^*_{z,.}$) ou bien selon les méthodes, l'écart entre les valeurs de diffusivité ne dépasse pas 5%.

Le milieu homogénéisé est donc homogène et isotrope. On peut donc considérer que le macroélément de référence K est un *Volume Élémentaire Représentatif* (VER) de pâte de ciment.

Pour $\rho = 0.1$, l'évolution des coefficients D^* en fonction de Ω^p est visible, surtout pour le coefficient $D^*_{y,m}$ qui atteint un écart de 9% avec $D^*_{y,p}$ dans le cas de la méthode Éléments Finis et de 15% pour la méthode Galerkin discontinue. On attribue cette évolution à un problème d'approximation des flux numériques : l'erreur dans le calcul du flux sur la cellule de référence $\hat{K}$ se répercute et se multiplie avec le nombre de macroéléments p.

Dans le cas $\rho = 0$, le calcul du flux s'appuie sur les conditions aux limites des problèmes de cellules, qui sont connues explicitement. Pour $\rho = 0.1$, le flux est calculé à partir des valeurs numériques des fonctions de bases locales Φ^i_K. Si la discrétisation de la couronne $\hat{K} \setminus K$ est trop grossière, le flux n'est pas approximé correctement. Quel axe est le plus influencé par cette erreur d'approximation ne dépend alors que de la position des inclusions de la couronne, qui conditionnent la génération du maillage $T_h(\hat{K})$.

On note de plus que les valeurs calculées par le biais de la méthode de Galerkin discontinue sont systématiquement inférieures à leurs équivalentes par la méthode Éléments Finis. On a vu à l'Exemple 7.2 que la méthode de Galerkin discontinue permet généralement d'obtenir une solution plus lisse, où l'erreur de résolution est répartie au sein du domaine au lieu d'être concentrée le long des frontières des macroéléments. Pour cette raison, on conjecture que les valeurs de diffusivités de la méthode de Galerkin discontinue sont plus justes, plus proches de la réalité, que leurs contreparties Éléments Finis. Faute de simulations ou d'expérimentations de référence à l'heure actuelle, il n'est cependant pas possible d'infirmer ou de confirmer ce point.

Méthode Éléments Finis, $\rho = 0$.

p	$D^*_{x,m}$	$D^*_{x,p}$	$D^*_{y,m}$	$D^*_{y,p}$	$D^*_{z,m}$	$D^*_{z,p}$
5	2097.46	2035.23	2071.56	2082.11	2054.37	2068.02
10	2096.89	2034.67	2071.00	2081.54	2053.80	2067.45
20	2094.57	2032.42	2068.71	2079.24	2051.54	2065.17

Méthode Éléments Finis, $\rho = 0.1$.

p	$D^*_{x,m}$	$D^*_{x,p}$	$D^*_{y,m}$	$D^*_{y,p}$	$D^*_{z,m}$	$D^*_{z,p}$
5	2116.06	2067.39	2105.23	2106.16	2080.85	2093.43
10	2120.40	2066.18	2166.11	2099.90	2081.80	2090.98
20	2129.40	2063.68	2292.32	2086.93	2083.78	2085.90

Méthode Galerkin discontinue, $\rho = 0$.

p	$D^*_{x,m}$	$D^*_{x,p}$	$D^*_{y,m}$	$D^*_{y,p}$	$D^*_{z,m}$	$D^*_{z,p}$
5	2005.43	1943.91	2043.39	1946.57	1965.16	1985.57
10	2004.91	1943.38	2042.86	1946.04	1964.65	1985.07
20	2003.89	1942.40	2039.72	1942.90	1962.39	1982.80

Méthode Galerkin discontinue, $\rho = 0.1$.

p	$D^*_{x,m}$	$D^*_{x,p}$	$D^*_{y,m}$	$D^*_{y,p}$	$D^*_{z,m}$	$D^*_{z,p}$
5	2022.63	1978.32	2074.75	1976.12	1990.09	2005.36
10	2028.75	1975.82	2136.04	1968.25	1991.22	2002.12
20	2035.91	1972.93	2242.98	1956.03	1994.96	1996.40

TABLE 7.3 – Exemple 7.5 : diffusivités équivalentes D^* calculées selon les deux directions des trois axes de l'espace pour différents domaines Ω^p. Les valeurs sont exprimées en $\mu\text{m}^2.\text{s}^{-1}$.

7.2.3 Application à un échantillon aléatoire de pâte de ciment.

7.2.3.1 Description du milieu.

On étudie ici un milieu semblable à celui décrit à la section §7.2.2, puisqu'il s'agit d'un échantillon de pâte de ciment de type *CEMI* [43, 42, 45]. Cependant, contrairement à l'Exemple 7.5 le milieu n'est pas construit périodiquement à partir d'un unique macroélément, mais généré aléatoirement par le biais du module COMBS (cf. §4.3.1).

Il s'agit d'un exemple représentatif des différents ordres de grandeur que l'on rencontre lors de simulations réalistes sur les matériaux cimentaires. On travaille en effet avec plusieurs dizaines de milliers d'inclusions qui provoquent des sauts de diffusivité d'un facteur 1×10^{11} au sein du milieu. La discrétisation complète du domaine Ω demande près de 100 millions de tétrahèdres, ce qui exclut toute résolution d'un problème de diffusion par une méthode numérique classique.

Exemple 7.6 *On considère un domaine cubique $K = [0,L]^3$, où on a posé $L = 250\mu$m. Cet échantillon de pâte cimentaire occupe donc un volume de $15.625 \times 10^6 \mu\text{m}^3$.*

La structure de ce milieu est semblable à celle de l'Exemple 7.5. Le milieu est donc composé d'une matrice, modélisant la porosité capillaire, de diffusivité $D_0 = 2.24 \times 10^{-9}\text{m}^2.\text{s}^{-1}$, qui intègre des inclusions sphériques modélisant les composés principaux de la pâte de ciment :

- *Résidus anhydres, portlandite, ettringite et hydrogrenats de diffusivité $D_a = 1 \times 10^{-20}\text{m}^2.\text{s}^{-1}$.*
- Low Density $C-S-H$ *de diffusivité $D_l = 9 \times 10^{-12}\text{m}^2.\text{s}^{-1}$.*
- High Density $C-S-H$ *de diffusivité $D_h = 1 \times 10^{-12}\text{m}^2.\text{s}^{-1}$.*

Au contraire de l'Exemple 7.5, cet échantillon de pâte cimentaire ne comporte pas d'inclusions multi-couches.

Environ 42000 inclusions sont placées au sein du domaine Ω. On présente à la Figure 7.13 la distribution de la taille des inclusions, les barres illustrant la répartition des inclusions sur les différentes phases considérées.

7.2.3.2 Partie locale de la chaîne de calcul.

On prépare le milieu Ω afin de pouvoir y résoudre des problèmes par les méthodes multi-échelles Q_1/*VFDiam* et GD/*VFDiam*. On divise Ω en 10 macroéléments dans chaque direction de l'espace, $\mathscr{D} = (10,10,10)$. Le maillage grossier $T_H(\Omega)$ correspondant comprend donc 1000 macroéléments K.

Pour des raisons techniques décrites à la Remarque 7.4, on doit s'assurer ici que les inclusions ne touchent pas les faces des macroéléments K. Seules 60% des inclusions générées sont donc placées au sein du domaine Ω, soit en moyenne 25 par macroélément. Deux valeurs du taux de sur-échantillonnage ρ sont considérées : $\rho = 0$ et $\rho = 0.2$.

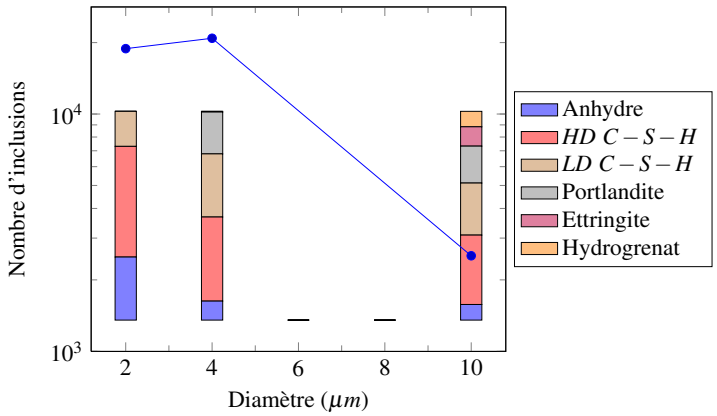


FIGURE 7.13 – Exemple 7.6 : distribution de la taille des inclusions dans le domaine Ω. Les barres présentent la répartition des inclusions sur les différentes phases considérées.

Afin de pouvoir utiliser la méthode GD/*VFDiam*, chaque face F du maillage grossier K doit disposer d'une unique discrétisation $T_h(F)$, indépendamment de la cellule considérée. En d'autre termes, si F est la face commune aux macroélements K^+ et K^-, les restrictions à F des maillages $T_h(\hat{K}^+)$ et $T_h(\hat{K}^-)$ doivent être identiques. On dit que les maillages $T_h(\hat{K}^+)$ et $T_h(\hat{K}^-)$ coïncident sur F.

Cette problématique est détaillée à la section §6.3, qui présente notamment l'Algorithme 6.9 utilisé pour construire les maillages de cellules coïncidents. On discrétise tout d'abord l'ensemble des faces Fde $T_H(\Omega)$, en imposant un rapport $\alpha = h/H$ valant $1/20$. Le maillage $T_h^{\mathscr{D}}(\Omega)$ de référence $2D$ ainsi obtenu comporte près de trois millions de faces.

Dans un second temps, et pour chaque valeur de ρ, on construit et on discrétise l'ensemble des cellules $\hat{K}$, en imposant la discrétisation issue de $T_h^{\mathscr{D}}(\Omega)$ sur les faces ∂K correspondantes. En moyenne, chaque maillage $T_h(\hat{K})$ possède 100000 éléments tétrahèdriques ($\rho = 0$) ou 268000 éléments ($\rho = 0.2$). On présente à la Figure 7.14 un exemple de maillage de cellules, dans le cas où $\rho = 0$ (à gauche) et $\rho = 0.2$ (à droite).

Remarque 7.4 *On doit s'assurer ici que les inclusions ne touchent pas les faces du macroélément K. En effet, les surfaces des inclusions sont maillées au sein de chaque cellule par un algorithme 2DD spécifique, généralement* BLSURF. *L'intersection d'une inclusion et de la face du macroélément se voit donc assigner deux algorithmes de discrétisation : d'une part la discrétisation imposée par le maillage 2DD de référence $T_h^{\mathscr{D}}(\Omega)$ et le mailleur 2DD de la cellule d'autre part. Ces deux algorithmes se perturbent l'un l'autre et le maillage obtenu ne coïncide pas sur les faces du macroélément K.*

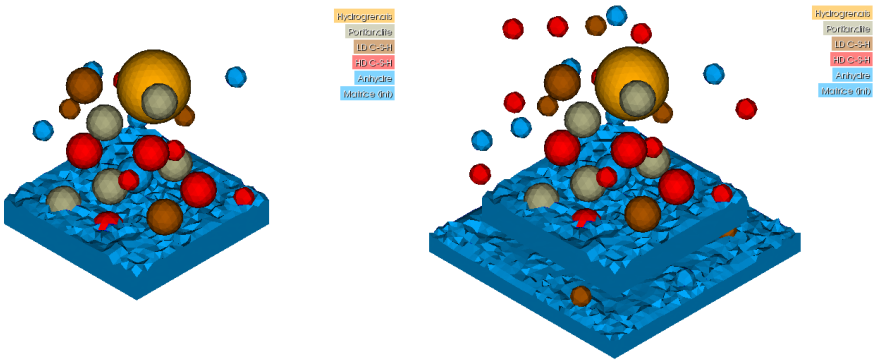


FIGURE 7.14 – Exemple 7.6 : visualisation *via SALOME* du maillage de la cellule $\hat{K} = (1,2,3)$ pour $\rho = 0$ (gauche) et $\rho = 0.2$ (droite). Le maillage de la matrice a été rendue invisible à partir d'une certaine hauteur afin de faire apparaître les inclusions. Chaque couleur correspond à une phase différente.

Une fois les maillages générés, les problèmes de cellules ont été résolus sur un cluster de calcul nommé *eris*. Il est composé de deux serveurs contenant chacun quatre cartes-mères. Une carte-mère contenant deux processeurs quadricoeurs *Intel(R) Xeon(R) X5667* (cadence 3GHz, 48Go de mémoire vive par carte), on dispose au total de 64 cœurs de calcul. 40 d'entre eux ont été utilisés pour cette opération. À chaque cellule ont été attribués 2 (cas $\rho = 0$) ou 4 cœurs de calcul (cas $\rho = 0.2$). Selon les cas, 20 ou 10 cellules ont donc été traitées de front.

On présente à la Figure 7.15 les temps de résolution des problèmes locaux en seconde, pour les cas $\rho = 0$ () et $\rho = 0.2$ (). Le temps moyen nécessaire pour résoudre un problème local est respectivement de 19s et 18s, auquel il faut ajouter 1s environ pour le traitement et le formatage des résultats. Construire la base locale et calculer les matrices locales demandent en moyenne 56s, respectivement 61s, portant à 216s, respectivement 213s, le temps total des opérations liés à une cellule.

Compte tenu des capacités de calculs disponibles, l'ensemble des cellules a donc été traité en $3h00$ pour le cas $\rho = 0$, et $5h55$ pour le cas $\rho = 0.2$, dont environ $2h15$, respectivement $4h12$, consacré à la résolution des 8000 problèmes de cellules.

7.2.3.3 Résolution de problèmes grossiers.

On résout sur Ω les trois problèmes d'homogénéisation précédemment décrit à la section §7.2.1.2 sous les noms des Exemples 7.4.1, 7.4.2 et 7.4.3. Ce problème est définit, pour la direction x, par :

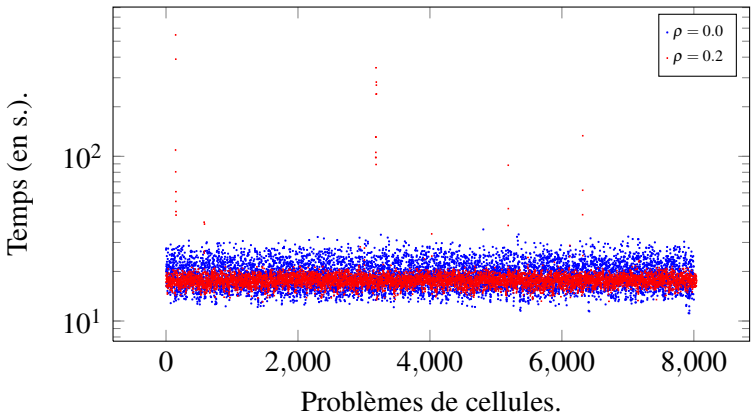


FIGURE 7.15 – Exemple 7.6 : temps de résolution des problèmes locaux en seconde, pour les cas $\rho = 0$ (·) et $\rho = 0.2$ (). Les temps moyens sont respectivement de $19s$. et $18s$.

Exemple 7.6.1

$$\begin{cases} -\nabla(D\nabla C) &= 0 & \text{dans } \Omega^p, \\ C &= 1 & \text{sur } \Gamma^p_{x,m}, \\ C &= 0 & \text{sur } \Gamma^p_{x,p}, \\ D\nabla C \cdot \mathbf{n} &= 0 & \text{sur les autres faces,} \end{cases} \tag{7.11}$$

où $\mathbf{n}$ *désigne le vecteur normal extérieur à* Ω^p *et où on a noté* $\Gamma^p_{x,m}$ *et* $\Gamma^p_{x,p}$ *les faces du domaine* Ω^p *pour lesquelles on a* $\{x = 0\}$ *et* $\{x = pL\}$ *respectivement. On définit de même les faces* $\Gamma^p_{y,m}$, $\Gamma^p_{y,p}$, $\Gamma^p_{z,m}$, $\Gamma^p_{z,p}$.

La Figure 7.16 visualise la solution fine $C_{H,h}$ du problème (7.11), pour la méthode Éléments Finis sans recouvrement ($\rho = 0$). On a ici reconstruit la solution fine sur la portion $\{100 \leq y,z \leq 125\}$ du domaine Ω. Ce volume correspond aux macroéléments numérotés $(i,4,4)_{0\leq i<10}$.

À gauche, a représenté la solution fine proprement dite, sur laquelle on a fait ressortir la discrétisation du milieu. À droite, on présente les surfaces d'iso-valeurs de $C_{H,h}$, qui illustent l'influence des inclusions sur la diffusion au sein du milieu.. Les couleurs s'étalonnent du rouge (en haut $x = 0$, $C_{H,h} = 1$) au bleu (en bas $x = 1$, $C_{H,h} = 0$).

7.2.3.4 Calcul de la diffusivité équivalente.

Par le biais des équations (7.8) et (7.9), on construit six diffusivités équivalentes à partir des solutions des problèmes d'homogénéisation. On considère dans la littérature que la pâte cimentaire se comporte comme un matériau homogène isotrope [42, 43]. Les diffusivités équivalentes D^* calculées ici doivent donc l'être aussi.

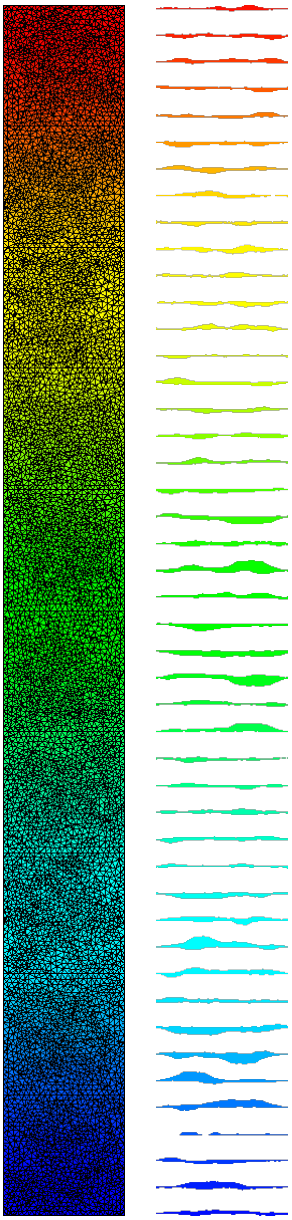


FIGURE 7.16 – Exemple 7.6.1 : visualisation de la solution fine $C_{H,h}$ (méthode Éléments Finis, $\rho = 0$) sur les macroéléments numérotés $(i,4,4)_{0\leq i<10}$, c'est-à-dire sur la portion $\{100 \leq y,z \leq 125\}$ du domaine Ω. À gauche, on présente la solution fine proprement dite, sur laquelle on a fait ressortir la discrétisation du milieu. À droite, on présente les isosurfaces de $C_{H,h}$. Les couleurs s'étalonnent du rouge (en haut $x = 0$, $C_{H,h} = 1$) au bleu (en bas $x = 1$, $C_{H,h} = 0$).

On présente au Tableau 7.4 ces valeurs de diffusivité pour les deux méthodes d'Éléments Finis et de Galerkin discontinue et pour les deux valeurs de ρ considérées : $\rho = 0$ et $\rho = 0.1$.

Dans le cas où on n'utilise pas de sur-échantillonnage ($\rho = 0$), les diffusivités calculées par la méthode d'Éléments Finis sont très proches les unes des autres, l'écart entre les six valeurs étant de 1.5%. Dans le cas où ρ est non nul, l'écart entre les valeurs diminue encore, pour ne pas dépasser 0.6%. Le milieu homogénéisé est donc homogène et isotrope et on peut considérer que le domaine Ω est un *Volume Élémentaire Représentatif* (VER) de pâte de ciment.

La méthode de Galerkin discontinue donne des résultats moins homogènes. Pour chaque axe, l'écart selon les directions ($D^*_{.,m}$ et $D^*_{.,p}$) est très faible, de l'ordre de 0.9%, si ρ est nul, et 0.5% sinon. Il y a cependant une disparité d'environ 14% entre les axes, un pourcentage trop élévé pour qu'on considère le milieu homogénéisé isotrope. Selon la méthode de Galerkin discontinue, Ω n'est donc pas un VER.

On peut attribuer cette différence d'homogénéité entre les deux méthodes à la suppression des inclusions coupant les frontières des macroéléments K (cf. Remarque 7.4). Cette opération modifie radicalement le domaine Ω aux bords des macroéléments, mais laisse leurs centres intactes. Cette suppression influence chaque axe de l'espace selon des degrés divers, en fonction de la position des inclusions supprimées. Quel axe est le plus influencé ne dépend alors que de la part d'aléatoire dans la génération du domaine Ω par le module COMBS.

Au contraire de la méthode d'Éléments Finis, la méthode de Galerkin discontinue s'appuie sur de nombreux termes calculées sur les faces des macroéléments (termes de saut, de pénalité). Il est donc logique que la solution du problème, et les coefficients équivalents correspondants, soient plus influencés par une modification artificielle du milieu autour de ces faces.

On note également que les valeurs calculées par le biais de la méthode de Galerkin discontinue sont systématiquement inférieures à leurs équivalentes par la méthode Éléments Finis, un phénomène déjà observé à l'Exemple 7.5. Par le même argument soulevé à l'Exemple 7.5, on suppose que les valeurs de diffusivités de la méthode de Galerkin discontinue sont plus justes, plus proches de la réalité, que leurs contreparties Éléments Finis.

En effet, la méthode de Galerkin discontinue permet généralement d'obtenir une solution plus lisse (cf. Exemple 7.2), où l'erreur de résolution est répartie au sein du domaine au lieu d'être concentrée le long des frontières des macroéléments. Cependant, il n'est pas possible d'infirmer ou de confirmer cette hypothèse, faute de simulations ou d'expérimentations de référence à l'heure actuelle.

Éléments Finis : $\rho = 0$

	$D^*_{.,m}$	$D^*_{.,p}$
$D^*_{x,.}$	2074.3	2086.3
$D^*_{y,.}$	2075.0	2063.8
$D^*_{z,.}$	2073.7	2044.9

Galerkin discontinue : $\rho = 0$

	$D^*_{.,m}$	$D^*_{.,p}$
$D^*_{x,.}$	1913.6	1924.7
$D^*_{y,.}$	1787.0	1779.9
$D^*_{z,.}$	1703.7	1689.2

Éléments Finis : $\rho = 0.1$

	$D^*_{.,m}$	$D^*_{.,p}$
$D^*_{x,.}$	2119.9	2117.2
$D^*_{y,.}$	2127.6	2122.9
$D^*_{z,.}$	2121.8	2111.9

Galerkin discontinue, $\rho = 0.1$

	$D^*_{.,m}$	$D^*_{.,p}$
$D^*_{x,.}$	2080.3	2070.5
$D^*_{y,.}$	1814.0	1822.6
$D^*_{z,.}$	1803.6	1805.7

TABLE 7.4 – Exemple 7.6 : diffusivités équivalentes D^* calculées selon les deux directions des trois axes de l'espace. Les valeurs sont exprimées en $\mu\text{m}^2.\text{s}^{-1}$.

Chapitre 8

Conclusions et perspectives.

On réalise dans ce chapitre le bilan des apports de ces travaux de thèse à la conception et à la mise en œuvre d'une chaîne de calcul multi-échelle pour l'homogénéisation des matériaux cimentaires.

On y reprend notamment les remarques ponctuelles que l'on a faites au cours des trois précédents chapitres sur les forces et faiblesses des choix de modélisation et d'implémentation effectués pendant ces travaux de thèse. Des pistes de travails sont proposées afin d'étendre les applications de ces travaux et, sur un plan plus pratique, d'améliorer la chaîne de calcul.

Cette conclusion s'articule autour de trois sections. On commence par présenter les réponses de ces travaux de thèse aux problématiques de la chaîne multi-échelle *SALOME-MPCube* proprement dite (§8.1), comme la gestion du parallélisme par exemple. On revient ensuite sur la modélisation des matériaux cimentaires et de leur dégradation (§8.2). Enfin la dernière section regroupe des considérations plus théoriques sur les méthodes multi-échelles (§8.3).

8.1 La chaîne multi-échelle *SALOME-MPCube*.

À l'issu de ces travaux de thèse, la chaîne de calcul multi-échelle *SALOME-MPCube* est fonctionnelle. Il suffit de disposer d'une description du milieu, précisant le nombre et la position des inclusions et la diffusivité des phases, pour que la chaîne de calcul réalise automatiquement l'ensemble des opérations nécessaires aux simulations multi-échelles : découpe du domaine en cellule $\hat{K}$, génération des maillages fins de cellules *via SALOME*, résolution des problèmes locaux par *MPCube*, etc.

Comme le montre le chapitre §7, il est relativement aisé de réaliser des simulations multi-échelles sur des matériaux cimentaires complexes, en utilisant l'une ou l'autre des méthodes Q_1/*VFDiam* ou GD/*VFDiam*. Avec l'Exemple 7.3 (cf. §7.1.3), on a assuré en particulier que la chaîne de calcul *SALOME-MPCube* supportait des simulations multi-échelles impliquant plus d'une centaine de milliers de macroéléments.

8.1.1 Parallélisme.

Comme on l'a expliqué la section §5.1, on peut espérer disposer de deux types de parallélisme au sein de la chaîne de calcul multi-échelle *SALOME-MPCube*. Il y a tout d'abord le parallélisme intrinsèque aux méthodes multi-échelles, dit *extra*-cellulaire : des opérations comme la discrétisation des cellules et la résolution des problèmes locaux peuvent être réalisées indépendamment d'une cellule à l'autre. On traite donc plusieurs cellules de front. En outre, *SALOME* et *MPCube* dispose de fonctionnalités permettant d'assigner plusieurs processeurs à une tâche unique. L'utilisation de ces mailleurs et solveurs parallèles constitue ce que l'on appelle le parallélisme *intra*-cellulaire.

La chaîne de calcul *SALOME-MPCube* a dès le début été pensée dans une vision de calcul parallèle. Autant que faire se peut, on a donc développé chaque module dans cette logique : répartition par cellule des données, implémentation $C++$ parallèle, etc. La Figure 5.1 récapitule les degrés de parallélisme, théoriques et actuellement implémentés, disponibles pour chaque module. Deux points en particulier sont à discuter.

En premier lieu, il est à noter que le traitement des cellules a été parallélisé au niveau *intra*-cellulaire et *extra*-cellulaire, en s'appuyant notamment sur des outils fournit par *MPCube* : MAKE_PAR.DATA et **Execute_parallel**. La résolution des problèmes locaux, la construction des fonctions de bases, le calcul des matrices locales associées et la mise en forme des résultats sont donc réalisés par jeux de plusieurs cellules, chaque cellule se voyant assigner un ou plusieurs cœurs de calcul.

Il s'agit là d'une étape très coûteuse de la méthode multi-échelle qui devient très vite rédhibitoire sans une implémentation parallèle adaptée. Les possibilités de calculs sont alors limités aux ordres de grandeurs présentés au chapitre §4. Au contraire, cette implémentation parallèle est très robuste et permet de réaliser rapidement des simulations impliquant un grand nombre de maillages (cf. l'Exemple 7.3 en §7.1.3) ou des maillages importants (cf. l'Exemple 7.6 en §7.2.3).

Au contraire de la résolution des problèmes locaux, la génération des maillages apparaît à la Figure 5.1 comme une tâche séquentielle, alors qu'on serait en droit d'attendre une implémentation profitant des deux parallélismes *intra*-cellulaire et *extra*-cellulaire.

SALOME disposant d'un mailleur parallèle (GHS3D PARALLEL) pouvant tirer partie de la disponibilité de plusieurs processeurs, il est possible de mettre en œuvre une forme de parallélisme *intra*-cellulaire. Cependant, cette fonctionnalité n'est indispensable que pour créer de très gros maillages, dépassant la dizaine de millions de volumes. Dans le contexte des méthodes multi-échelles, où on travaille par principe sur une série de problèmes de tailles raisonnables, cette forme de parallélisme n'a donc que peu d'intérêt.

En revanche, il serait plus utile de développer le parallélisme *extra*-cellulaire de la génération des maillages. Cette parallélisation est déjà partiellement réalisée, dans la mesure où on construit chaque maillage de cellule indépendamment des autres (cf. §6). La construction s'effectue cependant séquentiellement, c'est-à-dire une cellule après l'autre sur un seul cœur de calcul. En effet, travailler de front sur plusieurs cellules demande de manipuler plusieurs instances de *SALOME*, ce qui est techniquement complexe car de nombreuses ressources sont partagées. La solution réside probablement dans l'utilisation du module d'interfaçage YACS qui permet de lancer et de coupler plusieurs code de calculs. On lancerait ainsi une seule instance de *SALOME* qui piloterait les constructions paralléles des cellules.

Il est à noter que cette solution permettrait de profiter d'un parallélisme *extra*-cellulaire lors de la reconstruction de la solution fine $C_{H,h}$ car cette étape de la chaîne de calcul fait également appel à *SALOME*. On rencontre les même difficultés techniques empêchant la manipulation de plusieurs instances de *SALOME*, ce qui explique que cette étape s'effectue séquentiellement, c'est-à-dire une cellule après l'autre sur un seul cœur de calcul.

8.1.2 Génération automatique des maillages.

Au chapitre §6, on a présenté les méthodes permettant de construire automatiquement et efficacement les discrétisations $T_h(\hat{K})$ des cellules $\hat{K}$ du domaine de travail. Ces méthodes utilisent la plate-forme *SALOME*. On a pu ainsi construire avec un minimum d'effort les maillages nécessaires aux exemples présentés à la section §7.2, que ces maillages coïncident (Exemple 7.6) ou non (Exemple 7.4) sur les frontières des macroéléments K.

Outre les méthodes de discrétisations proprement dites, certaines fonctionnalités natives de *SALOME* ont été optimisées pour les besoins spécifiques des simulations multi-échelles, par exemple la coupe d'un «gros» objet par un ensemble de petits objets afin de construire la matrice du milieu. Le gain, en terme de temps de calcul, de ces optimisations est illustré par les très bons résultats de l'Algorithme 6.7.

La panoplie d'outils dont on dispose n'est cependant pas parfaite. La génération de maillages coïncidents en particulier nécessite certaines améliorations avant de constituer un outil véritablement efficace. À l'heure actuelle, on doit en effet assurer que les inclusions du milieu ne touchent pas les faces des macroéléments K pour que la méthode fonctionne correctement. La seule solution pour obtenir les maillages coïncidents nécessaires à l'utilisation de la méthode GD/*VFDiam*, est de supprimer purement et simplement les inclusions coupant les faces des macroéléments. Cette restriction limite donc fortement l'utilisation des maillages coïncidents lors de simulations réalistes de matériaux cimentaires.

L'origine de cette limitation a été détaillée à la Remarque 7.4. Les surfaces des inclusions sont maillées au sein de chaque cellule par un algorithme spécifique, gé-

néralement BLSURF, tandis que la discrétisation des faces des macroéléments est imposée par un maillage de référence (cf. §6.3). L'intersection de la surface d'une inclusion et de la face d'un macroélément se voit donc assigner deux algorithmes de discrétisation par *SALOME*, qui se perturbent l'un l'autre. Dans ces conditions, le comportement de *SALOME* n'est pas celui attendu, et les maillages obtenus ne coïncident pas sur les faces des macroéléments K.

On a envisagé deux pistes de travail afin de lever cette limitation. Faute de temps, elles n'ont cependant pas pu être étudiées.

La première idée, et la moins contraignante sur le plan de l'implémentation, consiste à manipuler les options des différents mailleurs de *SALOME* afin d'obtenir le résultat escompté. BLSURF notamment est un outil complexe, qui n'a pas été totalement maîtrisé au cours de ces travaux de thèse. Peut-être est-il possible d'imposer une règle de priorité aux différents mailleurs utilisés ?

Dans le cas contraire, il est fort probable que cette fonctionnalité apparaisse dans les prochaines versions si une demande argumentée est faite en ce sens auprès des équipes DISTENE chargées du développement de BLSURF [131, 103]. Le développement de BLSURF est en effet très actif et en lien direct avec celui de *SALOME*.

Il est également possible de résoudre ce problème indépendamment des outils de maillages utilisés. Il suffit pour cela de répartir, au sein de l'objet géométrique *SALOME* décrivant le macroélément, les surfaces des inclusions en fonction de leurs localisations. On distinguera ainsi les surfaces fixées, c'est-à-dire cises sur les faces du macroélément K, des surfaces libres placées à l'intérieur de K. Lors du maillage du macroélément, chaque groupe obéira à un mailleur distinct afin d'éviter les interférences : le maillage de référence pour les surfaces fixées et BLSURF pour les surfaces libres.

L'implémentation de cette solution peut paraître complexe au premier abord, mais il s'agit en fait d'une adaptation relativement simple de méthodes développées au cours de ces travaux de thèse. En effet, lors de la construction de la géométrie d'une cellule $\hat{K}$, les inclusions sont déjà triées en fonction de leurs localisations vis-à-vis des faces du macroélément. De plus, on manipule très souvent les surfaces des objets (cube du macroélément, inclusions) plutôt que les objets eux-même afin de réduire le temps de certaines opérations comme on le décrit à la section §6.2.3.

Dans ces conditions, un tri précis des différentes surfaces manipulées dans les méthodes actuelles devrait permettre de résoudre le problème. Cette opération de tri peut cependant être très longue au sein de *SALOME*, aussi il conviendra de l'utiliser à bon escient, afin de ne pas dégrader les performances globales de la méthode de génération des maillages.

8.2 Homogénéisation des matériaux cimentaires.

8.2.1 Homogénéisation numérique.

La chaîne de calcul *SALOME-MPCube* a été appliquée à des exemples de matériaux cimentaires, qui ont fait l'objet de la section §7.2. On a présenté notamment un échantillon aléatoire de pâte cimentaire (§7.2.3) qui est représentatif des différents ordres de grandeur que l'on rencontre en travaillant sur des matériaux cimentaires : plusieurs dizaines de milliers d'inclusions, des sauts de diffusivité d'un facteur 1×10^{11} et une discrétisation du domaine Ω en près de 100 millions de tétrahèdres.

Pour chacun de ces exemples, une fois le problème grossier résolu, on a pu reconstruire une solution à l'échelle fine sur tout ou partie du domaine. On a en outre calculé les diffusivités équivalentes D^* des matériaux, par le biais des formules (7.8) et (7.9) fondées sur les mesures expérimentales de diffusivités équivalentes.

Au chapitre §7, on a analysé qualitativement les résultats obtenus. En particulier, on a quantifié les écarts des diffusivités équivalentes correspondantes aux différentes directions de l'espace. En effet, on considère généralement dans la littérature [36, 52, 70] qu'un échantillon d'un matériau cimentaire doit être isotrope pour être considéré comme *représentatif*. L'écart entre les différentes diffusivités équivalentes numériques doit donc être faible.

Cependant, il n'a pas été possible de juger de l'impact de la méthode multi-échelle sur les valeurs de D^*. On ne disposait en effet pas de valeurs de référence, c'est-à-dire issues de simulations directes sur le domaine de travail Ω complet. Si une simulation directe sur l'échantillon de pâte cimentaire n'est pas réalisable, la discrétisation du domaine demandant près de 100 millions de tétrahèdres, on peut envisager de travailler sur des échantillons plus modestes.

On peut par exemple réutiliser les simulations réalisées en collaboration avec le laboratoire LECBA du CEA Saclay sur la dégradation de la pâte cimentaire [70]. Pour réaliser les simulations multi-échelles correspondantes, il est cependant nécessaire de résoudre les problèmes liés à la génération des maillages (cf. §8.1.2) et d'améliorer la modélisation des matériaux cimentaires (cf. §8.2.2).

8.2.2 Modélisation des matériaux cimentaires.

Comme on l'a décrit à la section §3.6, les matériaux cimentaires sont modélisés dans ces travaux de thèse par une collection d'inclusions, représentant les microstructures caractéristiques du milieu, que l'on enchâsse au sein d'un matériau homogène, la matrice. La forme de ces inclusions dépend de la phase que l'on souhaire représenté. Ainsi la portlandite s'organise en plaquettes hexagonales, tandis que l'ettringite forme des structures en aiguilles [159]. Malgré cette diversité de formes, on s'est restreint ici aux seules inclusions sphériques, que ce soit au sein

de la maquette Python (Partie II) comme de la chaîne de calcul *SALOME-MPCube* (Partie III).

Dans le cas de la maquette Python, cette contrainte provient d'une limitation du module COMBS, développé par Erwan ADAM, que l'on utilise pour placer aléatoirement les inclusions dans le domaine de travail. En effet, la première version de ce module, la seule disponible lors de l'implémentation de la maquette Python, ne permettait que de placer des inclusions sphériques à une ou plusieurs couches [21]. Les dernières versions de COMBS, en faisant appel à la plate-forme *SALOME*, permettent maintenant de placer des formes quelconques en deux et trois dimensions [70].

L'algorithme de placement des inclusions n'est plus limitant dans le cas de la chaîne de calcul *SALOME-MPCube*. La restriction des inclusions aux seules formes sphériques est ici liée à la construction des maillages de cellules $T_h(\hat{K})$. Dans le chapitre §6, on décrit les méthodes que l'on a développées pour construire automatiquement et efficacement ces maillages au sein de *SALOME*.

La section §6.2.4 en particulier présente le problème que pose une inclusion qui touche les parois de la cellule $\hat{K}$ en un unique point, appelé *points de tangence*. Le traitement de ce cas de figure passe par l'ajout de ces points de tangence à l'objet *SALOME* modélisant la géométrie de la cellule, ce qui demande de les connaître explicitement. À notre connaissance, il n'existe pas de méthode simple pour connaître, *via SALOME*, les points de tangences entre une inclusion de forme quelconque et un plan donné. La prise en charge des points de tangence, et par extension la génération automatique des maillages de cellules, a donc été restreint aux seules inclusions sphériques, pour lesquelles il est possible de calculer explicitement les éventuels points de tangence.

Si l'on souhaite décrire au plus juste la réalité des microstructures composant les matériaux cimentaires, il est nécessaire de s'affranchir de cette contrainte sur la forme des inclusions. On peut envisager deux pistes pour y arriver.

En premier lieu, on peut simplement désactiver le traitement automatique des points de tangence pour les inclusions non sphériques. De cette manière, on peut toujours demander à *SALOME* de discrétiser automatiquement les cellules sans points de tangence, tandis que l'utilisateur construit manuellement les maillages sur les cellules problématiques. Cette option a un intérêt limité, car elle ne peut être appliquée qu'aux domaines où les inclusions non sphériques ont peu ou pas de points de tangence, sous peine de rendre rédhibitoire le travail de l'utilisateur.

Une seconde solution, plus pérenne, consiste à développer une méthode, sous *SALOME*, pour déterminer les points de tangences entre une inclusion de forme quelconque et un plan donné. Il s'agit d'un problème de géométrie complexe et il n'est pas sûr que la plate-forme *SALOME* actuelle dispose des fonctionnalités nécessaires à cette tâche. Il est à noter cependant que le développement, la correction et l'amélioration de *SALOME* sont très actifs. En effet, si les versions 4.1.3 puis

5.1.4 ont été utilisées lors de cette thèse, une version 6.3 est attendue pour Mai 2011. On peut donc espérer que ces futurs développements fournissent, si ce n'est la solution complète au problème, au moins les outils dont on aura besoin pour la construire.

On peut envisager d'autres apports à la modélisation des matériaux cimentaires, par exemple en autorisant les inclusions d'une même phase à s'agréger en blocs. On peut également privilégier certaines directions lors de la génération des échantillons : placement des inclusions en lignes ou en plans, orientation des inclusions non sphériques le long d'un axe, etc.

8.2.3 Dégradation des matériaux cimentaires.

La section §3.6.2 présente une modélisation de la dégradation des matériaux cimentaires par une solution agressive, généralement de l'eau. Cette dégradation se traduit soit par un changement du champ de diffusivité (modification des caractéristiques physiques de la phase attaquée), soir par une extension du domaine de travail (une phase non-diffusive est dissoute et remplacée par du vide interstitielle).

Les solutions techniques permettant de générer efficacement les discrétisations associées aux étapes d'un processus de dégradation sont présentées à la section §6.4. Elles ont été mises en œuvre dans le cadre d'une collaboration avec le laboratoire LECBA du CEA Saclay [70]. Cette collaboration s'est effectuée dans un contexte de simulation classique, c'est-à-dire que les problèmes de diffusion ont été résolus par une méthode Volumes Finis *VFDiam* directe et non par une méthode multi-échelle.

Certains problèmes restent à résoudre avant de réaliser des simulations multi-échelles liées à la dégradation des matériaux cimentaires. Pour étudier le processus de dégradation, on travaille sur des domaines particulier où certaines zones, non diffusives, sont exclues du problème. On peut le voir par exemple sur le domaine Ω^0 présenté à la Figure 6.12.

Or les méthodes multi-échelles développées au cours de ces travaux de thèse nécessitent des macroéléments et des cellules rectangulaires ou parallélépipédiques. Les conditions aux limites des problèmes de cellules n'ont de sens que pour des frontières en arêtes (cas $2D$) ou rectangulaires (cas $3D$). En outre, la technique de sur-échantillonnage s'appuie sur les valeurs des fonctions $(\Psi^i_{\hat{K}})_{1\leq i\leq n}$, solutions des problèmes de cellules, en les sommets $(S_i)_{1\leq i\leq n}$ des macroéléments correspondants.

Avec les méthodes actuelles, on ne peut donc réaliser une simulation multi-échelle de dégradation des matériaux cimentaires qu'à la condition *sine qua non* où les zones non diffusives ne contiennent pas les sommets des macroéléments, qui sont aussi les points du maillage grossier $T_H(\Omega)$, et ne touchent aucune frontière des cellules $\hat{K}$. Les inclusions du milieu, diffusives ou non, possèdent cependant

une distribution spatiale très complexe qui rend le respect de cette condition fort improbable.

8.3 Aspects théoriques.

8.3.1 Les méthodes multi-échelles.

Ces travaux de thèse ont permis de développer deux méthodes multi-échelles couplant une méthode Volumes Finis *VFDiam* au niveau fin avec une méthode d'Éléments Finis (méthode Q_1/*VFDiam*) ou de Galerkin discontinue par pénalité intérieure (méthodeGD/*VFDiam*) au niveau grossier. L'implémentation efficace de ces deux méthodes a été réalisée au sein de la chaîne de calcul *SALOME-MPCube*. Plusieurs exemples, présentés aux sections §5.2 et §7.1, attestent de la qualification de cette implémentation et de la convergence des méthodes multi-échelles.

La convergence théorique de la méthode GD/*VFDiam* n'a pas été démontrée ici. Contrairement à la méthode Q_1/*VFDiam* [121], il n'existe pas à notre connaissance de preuve de cette convergence dans la littérature scientifique. Cette démonstration pourrait cependant s'appuyer sur les travaux de *Abdulle* [16] qui démontre la convergence d'une méthode multi-échelle basée sur une méthode de Galerkin discontinue. Il s'agit dans ce cas d'une méthode de Galerkin approchée (cf. §2.3) et non une méthode de Galerkin directe comme les méthodes Q_1/*VFDiam* et GD/*VFDiam*.

Augmenter les choix possibles du taux de sur-échantillonnage ρ a été un des arguments en faveur du passage d'une maquette Python séquentielle (Partie §II) pour une architecture parallèle plus robuste et plus puissante (Partie §III). L'idée était de disposer d'une marge de manoeuvre suffisante pour capturer les effets de couches limites aux sein des matériaux cimentaires, ce que la première implémentation ne pouvait faire.

L'influence de ρ sur les résutats des deux méthodes multi-échelles n'a pas été étudiée en détail au cours de cette thèse, faute de temps. Elle demanderait cependant une étude approfondie où plusieurs cas théoriques seraient résolus en variant les différents paramètres des méthodes multi-échelles : pas H, facteur $\alpha = h/H$, taux ρ et pénalité μ.

Dans cette thèse, on impose aux problèmes de cellules des conditions aux limites linéaires (cf. §3.2). D'autres possibilités existent, en particulier les conditions *oscillantes* [28, 120] qui donnent de très bons résultats. Le principe est d'utiliser la solution d'un problème de diffusion à la dimension $d-1$ comme conditions aux limites des problèmes de cellule en dimension d. Si on s'en est tenu aux conditions linéaires ici, c'est que les conditions oscillantes sont difficiles à mettre en œuvre, surtout dans un contexte $3D$. En effet, une utilisation brute des conditions oscillantes demande de résoudre 3 problèmes $2D$ et 6 problèmes $1D$ pour chaque pro-

blème de cellule. Soit, si on découpe le domaine en p macroéléments dans chaque direction, respectivement $48p^3$, $24p^3$ et $8p^3$ problèmes respectivement $1D$, $2D$ et $3D$.

Cependant, afin de pouvoir utiliser une méthode de Galerkin discontinue pour résoudre le problème grossier, on a développé des outils permettant de construire des maillages de cellules coincidents (cf. §6.3). Pour deux macroéléments voisins, de face commune F, la discrétisation $T_h(F)$ de F est alors identique, qu'elle provienne du maillage de l'un ou l'autre des macroéléments. Il devient alors inutile de résoudre deux fois les problèmes $2D$ sur F (une fois pour chaque macroélément). Avec des maillages coincidents, il suffit de résoudre un nombre beaucoup plus faible de problèmes, respectivement $6p(p+1)^2$, $12p^2(p+1)$ et $8p^3$ problèmes $1D$, $2D$ et $3D$ pour être précis. Dans ces conditions, l'utilisation des conditions oscillantes redevient compétitif.

8.3.2 Extensions du problème.

On s'est concentré pendant ces travaux de thèse sur un problème de diffusion stationnaire non convectif. Les outils de la chaîne de calcul multi-échelle *SALOME-MPCube* peuvent cependant être utilisés pour résoudre un problème d'évolution en temps.

En effet, la base de Galerkin, issue des solutions des problèmes locaux, ne dépend que des caractéristiques du domaine de travail Ω. Si la diffusivité du milieu est indépendante du temps, il suffit donc de la calculer une seule fois. On peut ensuite assembler et résoudre le problème grossier instationnaire un pas de temps après l'autre. Il est à noter cependant que c'est le problème grossier, et non les problèmes fins, qui doit vérifier les éventuelles conditions de stabilité du schéma numérique. Pour un schéma explicite en temps, la conditions *CFL* imposera donc une évolution très lente [50, 94].

D'autres extensions du problème sont envisageables, par exemple l'ajout d'un terme de convection ou de réaction au problème de diffusion. On peut également coupler différentes équations de diffusion afin d'étudier le transport de plusieurs espèces chimiques.

Quatrième partie

Annexes.

Annexe A

Schéma *VFDiam* en dimension 3.

On décrit dans cette annexe le schéma *VFDiam* [79] dans le cadre tri-dimensionnel, le cadre $2D$ ayant déjà fait l'objet d'une description précise à la section §3.2.3. On reprend d'ailleurs ici les notations de cette section.

On considère deux éléments adjacents k^+ et k^- du maillage fin $T_h(\hat{K})$, et on note $f = S_0S_1S_2$ leur face commune. On note G_f le barycentre de f. Les inconnues, respectivement u^+ et u^-, sont situées aux barycentres G^+ et G^- des éléments. On suppose la diffusivité A constante sur chaque élément, de valeurs respectives A^+ et A^-. On se reportera à la Figure A.1 pour une illustration des notations employées.

Comme dans le cadre $2D$, la méthode *VFDiam* s'appuie sur la construction d'un volume autour de la face f, que l'on appelle le *volume VFDiam*. Il s'agit de l'union des deux demi-élements diamants $k_f^+ = S_0S_1S_2G^+$ et $k_f^- = S_0S_1S_2G^-$, de volumes respectifs V^- et V^+. On introduit ensuite G_f, le barycentre de f, et les inconnues intermédiaires u_{S_i} localisées sur les noeuds S_i.

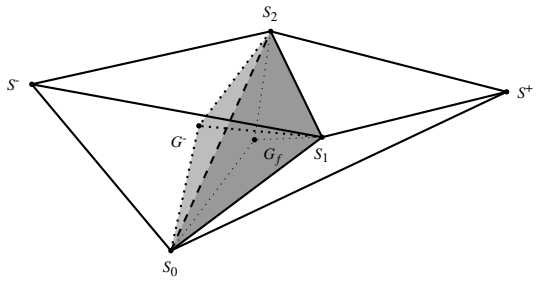


FIGURE A.1 – Un élément *VFDiam* est défini le long de $f = S_0S_1S_2$ (en gris foncé), la face commune des deux éléments adjacents $k^+ = S^+S_0S_1S_2$ et $k^- = S^-S_0S_1S_2$. Pour des raisons de lisibilité, on a ici représenté le seul demi-élement diamant $k_f^- = G^-S_0S_1S_2$ (en gris clair).

Sur le volume k_f^+, on a :

$$\begin{aligned}
< A\nabla u >_{k_f^+} &= \frac{1}{V^+}\int_{k_f^+} A\nabla u \\
&= \frac{A^+}{V^+}\int_{\partial k_f^+} u\,\mathbf{n} \\
&= \frac{A^+}{V^+}\left(\sum_{0\le i\le 2}\int_{G^+S_iS_{i+1}} u\,\mathbf{n} + \int_f u\,\mathbf{n}\right) \\
&= \frac{A^+}{V^+}\left(\sum_{0\le i\le 2}\int_{G^+S_iS_{i+1}} u\,\mathbf{n} + \sum_{0\le i\le 2}\int_{G_fS_iS_{i+1}} u\,\mathbf{n}\right)
\end{aligned}$$

où $\mathbf{n}$ désigne le vecteur normal unitaire extérieur à k_f^+, et où les sommets sont numérotés à un modulo 3 près. On a ici distribué la quantité $u\mathbf{n}$ sur chaque face de k_f^+, puis réparti cette même quantité sur les trois portions barycentriques de f. Sur chaque face, ou portion de face, on va maintenant approximer u par la moyenne des valeurs aux sommets correspondants.

Pour $0 \le i \le 2$, on note $\mathbf{n}_i^+$ le vecteur normal à la face $G^+S_iS_{i+1}$, extérieur à k_f^+ et de norme $\|G^+S_iS_{i+1}\|$. De même, on note $\mathbf{n}_f^+$ le vecteur normal à la face $f = S_0S_1S_2$, extérieur à k_f^+ et de norme l'aire de f. On a alors :

$$< A\nabla u >_{k_f^+} \approx \frac{A^+}{V^+}\left(\sum_{0\le i\le 2}\frac{u^+ + u_{S_i} + u_{S_{i+1}}}{3}\mathbf{n}_i^+ + \sum_{0\le i\le 2}\frac{u_{G_f} + u_{S_i} + u_{S_{i+1}}}{3}\frac{\mathbf{n}_f^+}{3}\right) \quad (A.1)$$

$$\approx \frac{A^+}{3V^+}\left(\left[\sum_{0\le i\le 2}\mathbf{n}_i^+\right]u^+ + \mathbf{n}_f^+ u_{G_f} + \sum_{0\le i\le 2}\left[\mathbf{n}_i^+ + \mathbf{n}_{i-1}^+ + \frac{2}{3}\mathbf{n}_f^+\right]u_{S_i}\right) \quad (A.2)$$

En tenant compte du fait que $\mathbf{n}_0^+ + \mathbf{n}_1^+ + \mathbf{n}_2^+ + \mathbf{n}_f^+ = \mathbf{0}$, et en posant $a^+ = A^+/(3V^+)$ et $\mathbf{t}_i^+ = \mathbf{n}_i^+ + \mathbf{n}_{i-1}^+ + \frac{2}{3}\mathbf{n}_f^+$ pour tout i, on obtient :

$$< A\nabla u >_{k_f^+} \approx a^+\left(\mathbf{n}_f^+(u_{G_f} - u^+) + \sum_{0\le i\le 2}\mathbf{t}_i^+ u_{S_i}\right) \quad (A.3)$$

De même, on peut calculer une formule similaire pour l'élément k_f^- :

$$< A\nabla u >_{k_f^-} \approx a^-\left(\mathbf{n}_f^-(u_{G_f} - u^-) + \sum_{0\le i\le 2}\mathbf{t}_i^- u_{S_i}\right) \quad (A.4)$$

On suppose maintenant que ces deux approximations vérifient la continuité des flux à travers la face f, c'est-à-dire que l'on a :

$$< A\nabla u >_{k_f^+}\mathbf{n}_f^+ + < A\nabla u >_{k_f^-}\mathbf{n}_f^- = 0 \quad (A.5)$$

On pose $\mathbf{n}_f = \mathbf{n}_f^+ = -\mathbf{n}_f^-$ pour alléger les notations. En rapportant (A.3) et (A.4) dans (A.5), il est possible d'obtenir une expression de u_{G_f} :

$$u_{G_f} = \frac{u^+(a^+\mathbf{n}_f)\cdot\mathbf{n}_f + u^-(a^-\mathbf{n}_f)\cdot\mathbf{n}_f + \sum_{0\leq i\leq 2} u_{S_i}\left[a^-\mathbf{t}_i^- - a^+\mathbf{t}_i^+\right]\cdot\mathbf{n}_f}{(a^+\mathbf{n}_f)\cdot\mathbf{n}_f + (a^-\mathbf{n}_f)\cdot\mathbf{n}_f} \tag{A.6}$$

En reportant (A.6) dans (A.3) ou (A.4), on dispose de l'approximation de flux suivante :

$$\begin{aligned} \tilde{L}_{k^+,f} &= < A\nabla u >_{k_f^+} \cdot\mathbf{n}_f^+ \\ \tilde{L}_{k^+,f} &= \frac{(a^-\mathbf{n}_f)\cdot\mathbf{n}_f(a^+\mathbf{n}_f)\cdot\mathbf{n}_f}{(a^+\mathbf{n}_f)\cdot\mathbf{n}_f + (a^-\mathbf{n}_f)\cdot\mathbf{n}_f}(u^- - u^+) \\ &\quad + \sum_{0\leq i\leq 2} \frac{(a^-\mathbf{n}_f)\cdot\mathbf{n}_f(a^+\mathbf{t}_i^+)\cdot\mathbf{n}_f + (a^+\mathbf{n}_f)\cdot\mathbf{n}_f(a^-\mathbf{t}_i^-)\cdot\mathbf{n}_f}{(a^+\mathbf{n}_f)\cdot\mathbf{n}_f + (a^-\mathbf{n}_f)\cdot\mathbf{n}_f} u_{S_i} \end{aligned} \tag{A.7}$$

Les autres aspects de la méthode *VFDiam*, interpolation par moindres carrés sur $\mathbb{S}_i$ et conditions aux limites de Dirichlet, sont identiques au cas $2D$. On se reportera donc à la section §3.2.3 pour leurs descriptions.

Annexe B

Codes sources et jeux de données de la chaîne de calcul *SALOME-MPCube*.

On présente dans cette annexe des codes sources Python, implémentations des méthodes *SALOME* permettant la construction et la discrétisation des cellules $\hat{K}$ du domaine Ω. Ces méthodes sont décrites au chapitre §6.

La seconde partie de cette annexe regroupe des exemple de fichiers *data*, les jeux de données permettant d'utiliser le code de calcul *MPCube*. §B.2.2§B.2.1.

B.1 Génération et opérations sur les maillages.

B.1.1 La méthode MAKEINTERSECTIONBYSHELLS.

La méthode MAKEINTERSECTIONBYSHELLS est l'implémentation Python de l'Algorithme 6.4. Elle permet d'obtenir la restriction d'une série de petits objets `inner_objects` à un objet de référence `outer_object`. S'ils ont été affectés, c'est-à-dire s'ils n'étaient que partiellement dans `outer_object`, les objets restreints sont appelés `intersected_inner_objects`. Dans le cas contraire, on les nomme `unaffected_inner_objects`.

Au sein de la méthode améliorée de construction de la géométrie de la cellule, décrite à la section §6.2.3, cette routine assure les étapes (Alg. 6.3 L.2) et (Alg. 6.3 L.6).

```
    # For each inner objects Obj, create the intersection of Obj with the outer object
    # Void elements are discarded, unaffected objects are put in a special list
    # Work for one shell outer object
4   # Work for multi shells inner_objects
    def MakeIntersectionByShells(outer_object, *inner_objects):

        shs_outer= geompy.SubShapeAll(outer_object, geompy.ShapeType["SHELL"])
        intersected_inner_objects=[]
9       unaffected_inner_objects=[]
```

```
    for obj in inner_objects:

        # Object shells
        shs_obj=geompy.SubShapeAll(obj, geompy.ShapeType["SHELL"])
        nb_shs_obj=len(shs_obj)

        # Part of object shells in the outer object
        shs_obj_in_outer= [ geompy.MakeCommon(sh, outer_object) for sh in shs_obj]

        # Sorting shells into null (i.e shells out of outer), affected (shells partially  in outer)
        # and unaffected (shells strictly in outer)
        affected_shs_obj_in_outer=[]
        is_obj_in_outer=False
        for j in range(nb_shs_obj):

            # Props 1 is area: the shell had to be non void.
            props= geompy.BasicProperties(shs_obj_in_outer[j])
            if props[1]>0:
                is_obj_in_outer= True

                # Looking for the result of the intersection
                props_ref= geompy.BasicProperties(shs_obj[j])
                diff= [ abs(props[k]- props_ref[k]) for k in range(len(props))]
                if max(diff)>1e-5:
                    # Shell has been "transformed" by the intersection.
                    affected_shs_obj_in_outer.append(shs_obj_in_outer[j])
                else:
                    unaffected_shs_obj_in_outer.append(shs_obj_in_outer[j])

        # Creating the intersection object
        if affected_shs_obj_in_outer:
            # Part of outer shells in obj
            shs_outer_in_obj = [geompy.MakeCommon(sh, obj) for sh in shs_outer]

            #Discarding void shells
            affected_shs_outer_in_obj=[]
            for sh in shs_outer_in_obj:
                props= geompy.BasicProperties(sh)
                if props[1]>0:
                    affected_shs_outer_in_obj.append(sh)

            # Intersection object
            cp_sh_intersected_obj= geompy.MakeShell(affected_shs_obj_in_outer+ affected_shs_outer_in_obj)
            shs_intersected_obj=geompy.SubShapeAll(cp_sh_intersected_obj, geompy.ShapeType["SHELL"])
            intersected_inner_objects.append(intersected_obj)

        else:
            if is_obj_in_outer:
                unaffected_inner_objects.append(obj)

    return [intersected_inner_objects, unaffected_inner_objects]
```

B.1.2 La méthode MAKECUTBYSHELLS.

La méthode MAKECUTBYSHELLS est l'implémentation Python de l'Algorithme 6.5. Elle permet de couper un objet de référence `outer_object` par un ensemble de petits objets `inner_objects` que l'on suppose disjoints. L'objet coupé est appelé `cut_object`.

Au sein de la méthode améliorée de construction de la géométrie de la cellule (cf. §6.2.3), cette routine assure les étapes (Alg. 6.3 L.9) et (Alg. 6.3 L.10).

```
# Create the cut of outer_object by the union of inner_objects
# Last modified: 08/03/2010
def MakeCutByShells(outer_object, *inner_objects):

    if not(inner_objects):
        return outer_object

    # Retrieving shells from outer_object
    shs_outer= geompy.SubShapeAll(outer_object, geompy.ShapeType["SHELL"])
```

```
    nb_shs_outer=len(shs_outer)

    # Working on each inner objects
    shs_over_outer=[]
    shs_in_outer=[]
    for obj in inner_objects:

        # Computing the potential cut of obj to outer shells
        is_obj_over_outer=False
        shs_holes_in_outer = [geompy.MakeCommon(sh, obj) for sh in shs_outer]
        for j in range(nb_shs_outer):
            props=geompy.BasicProperties(shs_holes_in_outer[j])

            # Props 1 is area: the shell had to be non void.
            if props[1]>0:
                shs_outer[j] = geompy.MakeCut(shs_outer[j], shs_holes_in_outer[j])
                is_obj_over_outer=True

        # Object shells
        shs_obj=geompy.SubShapeAll(obj, geompy.ShapeType["SHELL"])
        if len(shs_obj)!=1:
            raise "Error in MakeCutByShells: inner objects must have only one shell"
        sh_obj=shs_obj[0]

        # Part of object shell in the outer object
        sh_obj_in_outer= geompy.MakeCommon(sh_obj, outer_object)
        shs_obj_in_outer= geompy.SubShapeAll(sh_obj_in_outer, geompy.ShapeType["SHELL"])

        # Making the "patch of the holes"
        for sh in shs_obj_in_outer:
            props= geompy.BasicProperties(sh)
            if props[1]>0:
              # Shell is non-void: we change its orientation to "make it" a part of outer_shell
              sh_inv = geompy.ChangeOrientation(sh)
              if is_obj_over_outer:
                  # The obj shell is contiguous to the outer shell
                  shs_over_outer.append(sh_inv)
              else:
                  # We are strictly in the outer shell
                  shs_in_outer.append(sh_inv)

    # We create a solid from a list of shells
    all_shells_in_one=geompy.MakeShell(shs_outer+shs_over_outer)
    all_shells=geompy.SubShapeAll(all_shells_in_one, geompy.ShapeType["SHELL"])
    cut_object = geompy.MakeSolid(all_shells+shs_in_outer)
    return cut_object
```

B.1.3 La méthode MAKECONFORMTOTANGENTOBJECTS.

La méthode MAKECONFORMTOTANGENTOBJECTS ajoute les points de tangence des objets `tangent_objects` à un objet de référence `outer_object`. L'objet *SALOME* obtenu est appelé `partition`, en référence à la méthode MAKEPARTITION qui permet de le construire.

Elle assure l' étape (Alg. 6.7 L.5) de la méthode complète de discrétisation de la cellule (cf. §6.2.5).

On présente tout d'abord la sous-méthode GETCROSSLINES, qui est l'implémentation Python de l'Algorithme 6.6, puis la méthode MAKECONFORMTOTANGENTOBJECTS.

```
# Given a "tree" of lines, i.e a set of divisions of the original rectangle into rectangles.
# Given a point A.
# It computes the extremities of the 2 vertical and the 2 horizontal lines starting in A.
# Thoses xtremities belongs to the precedent lines.
# Only new lines are sent back, lines previously in the tree are ignored.
# Adapts the "tree" to reflect the new division.
# Last modified: 08/24/2010
def getCrossLines(point, lines_tree):
```

```
    # Check dimension
    if len(point)!=2:
        raise "It a 2D function! Point must have 2 coordinates."

    # Looking for the smallest rectangle containing point
    rectangle=lines_tree
    while len(rectangle)>1:
        # Looking for the sub-rectangle containing point
        center=[rectangle[1][0][2], rectangle[1][0][3]]
        if point==center:
            return []

        i= (point[0]>center[0])
        j= (point[1]>center[1])
        k= i + j*len(rectangle)/2
        rectangle=rectangle[k+1]

    # We are now in the good rectangle
    dimensions= rectangle[0]
    decX=[dimensions[0], dimensions[2]]
    if point[0] not in decX:
        decX=[decX[0], point[0], decX[1]]
    decY=[dimensions[1], dimensions[3]]
    if point[1] not in decY:
        decY=[decY[0], point[1], decY[1]]

    # Creation of new rectangles
    new_rectangles= [ [[decX[i],decY[j],decX[i+1],decY[j+1]]]
                        for j in range(len(decY)-1) for i in range(len(decX)-1) ]
    rectangle+=new_rectangles

    # Compute the lines starting in A as a list of 2 extremites Pi=(Xi,Yi) and Pj=(Xj,Yj).
    # A is either Pi or Pj.
    if len(decX)==2:
        lines=[ [[decX[0], point[1]] , [decX[1], point[1]]] ]
    elif len(decY)==2:
        lines=[ [[point[0], decY[0]] , [point[0], decY[1]]] ]
    else:
        linesX= [ [[decX[i], point[1]], [decX[i+1], point[1]]] for i in range(len(decX)-1)]
        linesY= [ [[point[0], decY[j]], [point[0], decY[j+1]]] for j in range(len(decY)-1)]
        lines=linesX+linesY

    return lines

# Given an outer object O
# Given tangent objects to O, with a SALOME vertex object on their geometrical tangent point.
# Partition the outer object with a set of lines crossing on the tangent points.
# Last modified: 08/24/2010
def makeConformToTangentObjects(self, outer_object, *tangent_objects):

    if not(tangent_objects):
        return outer_object

    all_lines=[]
    for obj in tangent_objects:

        cp_pts_on = geompy.GetShapesOnShapeAsCompound(outer_object, obj,
                                        geompy.ShapeType["VERTEX"], geompy.GEOM.ST_ON)
        pts_on = geompy.SubShapeAll(cp_pts_on, geompy.ShapeType["VERTEX"])

        for pt in pts_on:
            # Projection on the tangent face
            P= [coord for coord in geompy.PointCoordinates(pt)]
            num_face= self.GetFace(P)
            if num_face ==-1:
                raise "Error: tangent point not on a tangent face :/"
            [Q, ignored_indice, ignored_value]= getProjectionPoint(P, (num_face%6)/2)

            # Computing "cross" lines"
            lines=getCrossLines(Q, self.lines_tree[num_face])
            for l in lines:
                la= getCompletePoint(l[0], ignored_indice, ignored_value)
                pta=geompy.MakeVertex(la[0], la[1], la[2])
                lb= getCompletePoint(l[1], ignored_indice, ignored_value)
                ptb=geompy.MakeVertex(lb[0], lb[1], lb[2])
                salome_line=geompy.MakeLineTwoPnt(pta, ptb)
                all_lines.append(salome_line)

    if all_lines:
        partition=geompy.MakePartition([outer_object], all_lines,[],[],geompy.ShapeType["SOLID"])
    else:
        partition=outer_object
    return partition
```

B.2 Exemples de jeux de données *MPCube*.

On présente ici des exemples de fichiers *data*, les jeux de données permettant d'utiliser le code de calcul *MPCube*. Dans le cadre de la chaîne de calcul *SALOME-MPCube*, la plupart de ces jeux de données sont générés automatiquement, comme on l'explique à la section §5.1.2.

Les fichiers *data* sont des fichiers textes respectant certaines règles structurelles simples. Afin de déclarer un objet, on utilise une commande de la forme "type_objet nom". Les actions, par exemple résoudre un problème ou lire un fichier, présentent une forme similaire : "commande parametre1 ... parametreN". En utilisant des accolades { }, il est possible d'écrire des déclarations imbriquées. Enfin, les jeux de données ne sont pas sensibles à la casse et doivent se terminer par le mot-clé **fin**.

Tout texte compris entre deux # est un commentaire et n'est donc pas interprété par *MPCube*. Certains commentaires sont cependant interprétés lors de la préparation de la simulation. En effet, il ne faut pas oublier que ce jeu de données n'est pas exploitable en l'état, mais qu'il est nécessaire de faire appel à la commande MAKE_PAR.DATA afin de construire les fichiers nécessaires à la simulation *MPCube*. Les commentaires *#DEBUT DECOUPAGE ... FIN DECOUPAGE#* et *#DEBUT LECTURE ... FIN LECTURE#* du jeu de donnés permettent alors de préciser certains paramètres de la préparation, notamment les noms des fichiers temporaires construits.

B.2.1 Résolution d'un problème.

Le jeu de donnés présenté ici correspond à la résolution, par la méthode des Volumes Finis *VFDiam*, d'un problème de diffusion. Il permet de calculer la solution $\Psi^0_{\hat{K}}$ du premier problème d'une cellule (cf. §3.2). Cette cellule est appelée ici "omega0". Sa diffusivité adimensionnée vaut 2240, sauf en une série d'inclusion (groupe "IntInclusionsD0") où elle vaux 9. Il n'y a pas de sur-échantillonnage (ρ=0).

```
1  # PARALLEL OK 4 #
   Dimension 3
   Domaine omega0

   # DEBUT DECOUPAGE
6  Lire_med omega0 momega0 omega0.med
   Lire_fichier ssz.geo ;
   Ecrire_fichier omega0 omega0.geom
   Decouper omega0
   {
11    Partitionneur metis { Nb_parts 4 }
      Larg_joint 2
      Nom_Zones OMEGA0
   }
   Fin
16  FIN DECOUPAGE #

   # DEBUT MAILLAGE #
   Lire_fichier omega0 ./Travail/wX0/wY0/wZ0/omega0.geom
   Lire_fichier ./Travail/wX0/wY0/wZ0/ssz.geo ;
```

```
# FIN MAILLAGE #

# DEBUT LECTURE
Scatter ./Travail/wX0/wY0/wZ0/OMEGA0.Zones omega0
Lire_fichier ./Travail/wX0/wY0/wZ0/ssz_par.geo ;
 FIN LECTURE #

# Le milieu comporte deux zones:
    - IntFond, de diffusivité 2240.
    - IntInclusionD0, de diffusivité 9.
#
Milieu_Diffusion medium0
Lire medium0
{
  permeabilite Champ_Uniforme_Morceaux omega0 6
  {
    defaut              1  0  0     1  0     1
    IntFond          2240  0  0  2240  0  2240
    IntInclusionD0      9  0  0     9  0     9
  }
  porosite     Champ_Uniforme 1  1.
  porosite_constante
}

# On definit le schéma de résolution:
    - Librairie Petsc.
    - Preconditionner BOOMERAMG.
    - Solveur BICGSTAB.
#
Schema_Permanent scheme0
Lire scheme0
{
  solveur Solveur_Lineaire_Std
  {
    solveur NPPetsc
    {
      impr
      seuil 1e-7
      max_iter 20000
      solver NP_BICGSTAB NP_BOOMERAMG
    }
  }
  nb_iterations_maximal 2
}

# On specifie que l'on résout un problème de diffusion scalaire. #
Equation_Diffusion equation_diff0
Fermeture_Diffusion closure0
Lire closure0
{
  modele Scalaire_Lineaire
  {
    coefficient_inertiel_standard
  }
}
Associer equation_diff0 closure0

# Initialisation du probleme. #
CCFV ma_discretisation
Probleme_Diffusion probleme0
Associer probleme0 equation_diff0
Associer probleme0 omega0
Associer probleme0 scheme0
Associer probleme0 medium0
Discretiser probleme0 ma_discretisation

# Construction du probleme. #
Lire probleme0
{
  equation_diff0
  {
    # Choix de la méthode de Volumes Finis. #
    diffusion  { diamfv1 }
    conditions_initiales { inconnue Champ_fonc_txyz omega0 1 0 }

    # Le terme source, ici nul. #
    sources
    {
      analytique
      {
        expression Champ_fonc_txyz omega0 1 0
        quadrature_order 2
      }
    }
```

```
    # Les conditions aux limites. #
    conditions_limites
    {
        FAM_4_Bord_Xm Dirichlet_Valeur_Imposee Champ_front_fonc_txyz 1 (1-x)*(1-y)*(1-z)
        FAM_5_Bord_Xp Dirichlet_Valeur_Imposee Champ_front_fonc_txyz 1 (1-x)*(1-y)*(1-z)
        FAM_6_Bord_Ym Dirichlet_Valeur_Imposee Champ_front_fonc_txyz 1 (1-x)*(1-y)*(1-z)
        FAM_7_Bord_Yp Dirichlet_Valeur_Imposee Champ_front_fonc_txyz 1 (1-x)*(1-y)*(1-z)
        FAM_8_Bord_Zm Dirichlet_Valeur_Imposee Champ_front_fonc_txyz 1 (1-x)*(1-y)*(1-z)
        FAM_9_Bord_Zp Dirichlet_Valeur_Imposee Champ_front_fonc_txyz 1 (1-x)*(1-y)*(1-z)
    }
  }

  # Choix du format de sortie, ici le format lata. #
  # On souhaite avoir la solution et le flux.      #
  Postraitement
  {
    format lata
    Champs dt_post 1.
    {
      solution_inconnue elements
      flux_inconnue faces
    }
  }
}

# Résolution du problème. #
Resoudre probleme0
FIN
```

B.2.2 Renumérotation d'un maillage.

Le jeu de données suivant reporte les champs solutions contenues dans un fichier MED sur un maillage de référence. Pour que ce jeu de données fonctionne, il faut que le support des champs solutions soit identique, à la numérotation des éléments près, au maillage de référence. En sortie de ce problème, on obtient donc un fichier MED contenant les mêmes informations que le fichier d'origine, mais avec la numérotation du maillage de référence.

On utilise ce type de jeux de données lors de la mise en forme des résultats locaux, décrite à la section §5.1.5.

```
dimension 3

# DEBUT MAILLAGE #
Domaine pcellule3D_0_0_0
Lire_med work_directory /home/simulation/wX0/wY0/wZ0/ pcellule3D_0_0_0
         mcellule3D_0_0_0 /home/simulation/wX0/wY0/wZ0/cellule3D_0_0_0.med
Lire_fichier /home/simulation/wX0/wY0/wZ0/ssz.geo ;
# FIN MAILLAGE #

# DEBUT DECOUPAGE
Fin
FIN DECOUPAGE #
# DEBUT LECTURE
FIN LECTURE #

export Domaine cellule3D_0_0_0

Pbc_MED pb_renumerotation
Lire pb_renumerotation
 {
 /home/simulation/wX0/wY0/wZ0/cellule3D_0_0_0_Solution.med
 cellule3D_0_0_0
   {
   Postraitement
     {
     Domaine pcellule3D_0_0_0
     Format med
     Fichier /home/simulation/wX0/wY0/wZ0/cellule3D_0_0_0_projection
```

```
      Champs dt_post 0.0005
        {
        SolutionPhi_00 elem
        SolutionPhi_01 elem
        SolutionPhi_02 elem
        SolutionPhi_03 elem
        SolutionPhi_04 elem
        SolutionPhi_05 elem
        SolutionPhi_06 elem
        SolutionPhi_07 elem
        }
      }
    }
  }

FIN
```

Annexe C

Publications.

NETWORKS AND HETEROGENEOUS MEDIA
©American Institute of Mathematical Sciences
Volume **5**, Number **3**, September **2010**
doi:10.3934/nhm.2010.5.603
pp. **603–615**

APPLICATION OF A COUPLED FV/FE MULTISCALE METHOD TO CEMENT MEDIA

Thomas Abballe

CEA Saclay
DEN/DM2S/SFME/LSET Bat 454
91191 Gif-sur-Yvette Cedex , France

Grégoire Allaire

Conseiller Scientifique du DM2S – CEA Saclay
Centre de Mathématiques Appliquées – École Polytechnique
91128 Palaiseau Cedex, France

Éli Laucoin and Philippe Montarnal

CEA Saclay
DEN/DM2S/SFME/LSET Bat 454
91191 Gif-sur-Yvette Cedex , France

Abstract. We present here some results provided by a multiscale resolution method using both Finite Volumes and Finite Elements. This method is aimed at solving very large diffusion problems with highly oscillating coefficients. As an illustrative example, we simulate models of cement media, where very strong variations of diffusivity occur. As a by-product of our simulations, we compute the effective diffusivities of these media.After a short introduction, we present a theorical description of our method. Numerical experiments on a two dimensional cement paste are presented subsequently. The third section describes the implementation of our method in the calculus code *MPCube* and its application to a sample of mortar. Finally, we discuss strengths and weaknesses of our method, and present our future works on this topic.

1. Introduction.

1.1. **Heterogeneities in cement media.** Cement media are spatially very heterogeneous with a large variety of physical scales. Furthermore the diffusivity of the cement components varies with a very large contrast of several orders of magnitude.

For example, when working on the *concrete* scale, where the representative elementary volume (REV) is roughly a few centimeters wide, we consider that the cement paste is homogeneous and that aggregates (sand, gravel) are scattered in it, thus creating millimiters wide heterogeneities.

On the other hand, if we choose to work at the *cement paste* scale (REV width not exceeding 100μm), we have to take into account mineral microstructures, which sizes range from 1 to 30μm. This is therefore a multiscale problem.

2000 *Mathematics Subject Classification.* 74Q05, 65N30.

Key words and phrases. finite volume method, finite element method, multiscale method, cement media.

This work was done in relation with the GNR MOMAS (Modélisation Mathématique et Simulations numériques liées aux problèmes de gestion des déchets nucléaires) (PACEN/CNRS, ANDRA, BRGM, CEA, EDF, IRSN).

1.2. **Determination of the effective diffusivity.** Our goal is to compute the effective diffusivity at the concrete or cement paste scale. This topic has previously been addressed by various laboratories at the CEA, the French Atomic Energy Commission, both from physical and mathematical points of view. Strict analytical homogeneization [6] can be, in this case, very complex to use, chiefly because of the continuous variation and the random spatial distribution of the inclusions (depending of the scale: aggregates or mineral stages).

A direct numerical simulation in three dimensions is almost impossible because of the huge computational resources (memory, CPU time) required by fine meshes. Therefore, in order to achieve our goal, we have developed a specific multiscale resolution method using both Finite Volumes (FV) and Finite Elements (FE) and we present here its preliminary results.

Many multiscale numerical methods have recently been developed and it is almost impossible to quote all the relevant works, be it Finite Elements methods [12], [14], [18], [20], [22] or Finite Volumes methods [17], [19].

1.3. **A FV/FE Multiscale method.** Our method relies on the coupling of two grids: a coarse one and a fine one. The main idea is to build a Finite Element basis on the coarse grid from solutions computed on the fine one. In previous works on this subject [5],[18], the Finite Element method was used on both the fine and coarse grids, whereas, in our approach, the fine scale simulations are made using Finite Volumes. Thus we expect to increase the stability of the method in view of strong discontinuities and anisotropy of the studied media, and to keep the advantages of the multiscale Finite Element method (no geometric assumptions on the media, easy parallelization of the computation).

We solve a stationary diffusion equation to determine the concentration C of a single chemical species:

$$\left\{\begin{array}{rcll} -\nabla\cdot(D\,\nabla C) & = & \alpha & \text{in } \Omega, \\ C & = & f & \text{on } \Gamma_D, \\ D\,\nabla C\cdot n & = & g & \text{on } \Gamma_N, \end{array}\right. \tag{1}$$

where (Γ_D, Γ_N) is a partition of the boundary Γ of Ω.

2. **Description of the multiscale method.** In the sequel, Ω is either a rectangular domain (two dimensions) or a parallelepipedic one (three dimensions).

Algorithm 1 presents the various steps of our multiscale method we use for solving problem (1).

Algorithm 1 Main steps

1: Partition the domain into sub-domains with oversampling
2: Solve cell problems
3: Build the Finite Element basis
4: Solve the coarse problem
5: Compute the fine-scale solution

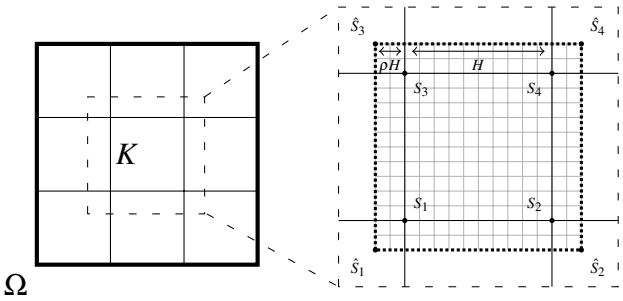


FIGURE 1. The domain Ω (bold line) is divided into rectangular macroelements K (solid line). The cell $\hat{K}$ (dotted line), built from K and a fraction ρ of its neighbours, is meshed finely (in thin grey).

2.1. **Decomposition into sub-domains with oversampling.** Assuming we know the diffusivity D on the fine scale, we divide the domain Ω into rectangular, respectively parallelepiped, macroelements K, thus creating a two-dimensional, respectively three-dimensional, regular coarse mesh. As shown in Figure 1, at the local scale, we define the cell $\hat{K}$ by enlarging the macroelement K with a fraction $\rho \geq 0$ of its neighbours. We call ρ the *oversampling rate*. Oversampling was introduced in [18] to improve the efficiency of multiscale finite element methods.

In order to save computing resources, the domain Ω is not globally meshed at the fine scale. The cells $\hat{K}$ are meshed individually, while ensuring that the meshes of two adjacent cells $\hat{K}_1$ and $\hat{K}_2$ are conforming on $\partial K_1 \cap \partial K_2$. Building smaller meshes for all cells than a large global mesh for the entire domain is of course much easier and efficient in terms of memory and CPU requirements. The fine scale meshes of $\hat{K}$ can be either structured or not.

2.2. **Solving cell problems.** With $n = 4$ (dimension 2) or $n = 8$ (dimension 3), let $(S_i)_{1\leq i\leq n}$ be the *vertices* of the macroelement K, and $(\hat{S}_i)_{1\leq i\leq n}$ the *vertices* of the cell $\hat{K}$.

On each cell $\hat{K}$ we have to solve a *cell problem* for each $1 \leq i \leq n$:

$$\begin{cases} -\nabla \cdot (D\, \nabla \Psi^i_{\hat{K}}) = 0 \ \text{ in } \ \hat{K}\,, \\ \Psi^i_{\hat{K}} = \beta^i \ \text{ on } \ \partial\hat{K}, \end{cases} \tag{2}$$

where β^i is a continuous linear function on each edge of $\partial\hat{K}$, with $\beta^i(\hat{S}_j) = \delta_{i,j}$, where δ stands for the Kronecker symbol.

For two-dimensional structured meshes it is possible to solve the cell problems by the *VF9 finite volume* method [16]. In most cases, as D is diagonal, this method is equivalent to the classical five points Finite Volume method. However, for more general cases we switched to the Diamond Finite Volume Method (*VFDiam*) for solving *cell problems*. First described in [11], *VFDiam* has been used by the CEA to work on media, like the cement ones, where strong anisotropies or/and contrasts of diffusivity occur. We now briefly recall the principles of *VFDiam*.

As shown in Figure 2 (left part), in the $2D$ case, we consider two adjacent cells where the unknowns, C^- and C^+ respectively, are located at their barycenter, and call $F = T_0T_1$ their common edge. We suppose D to be constant on each cell, equal to D^- and D^+ respectively.

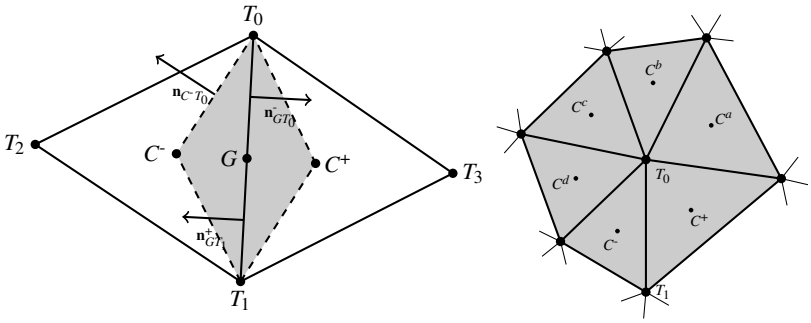


FIGURE 2. Left: a *VFDiam* element (in grey surrounded by a dashed line) is defined around the common face of two adjacent elements. Right: the set $\mathbb{T}_0$ (in grey) is defined as the collection of all cells which contain the node T_0.

We define the *VFDiam* volume as the union of the two half-diamond cells $T_0T_1C^-$ and $T_0T_1C^+$, respectively of volumes V^- and V^+. We then introduce G, the barycenter of F, and the intermediate unknowns C_{T_i} localised on T_i.

As a Finite Volume Method, the *VFDiam* method relies on the following approximation of the flux across F :

$$\int_F D\nabla C.\mathbf{n} = \frac{\mathbf{n}.\kappa^-\mathbf{n}\mathbf{n}.\kappa^+\mathbf{n}}{\mathbf{n}.\kappa^-\mathbf{n}+\mathbf{n}.\kappa^+\mathbf{n}}(C^+ - C^-) + \sum_{k=0}^{1}\left[\frac{\mathbf{n}.\kappa^+\mathbf{r}^+_{\mathbf{k}}\mathbf{n}.\kappa^-\mathbf{n}+\mathbf{n}.\kappa^-\mathbf{r}^-_{\mathbf{k}}\mathbf{n}.\kappa^+\mathbf{n}}{\mathbf{n}.\kappa^-\mathbf{n}+\mathbf{n}.\kappa^+\mathbf{n}}\right] C_{T_k} \tag{3}$$

where:

- $\mathbf{n}$ is the normal vector of F, oriented from C^- to C^+ and of norm $\|F\|$.
- $\kappa^- = \frac{D^-}{2V^-}$, respectively $\kappa^+ = \frac{D^+}{2V^+}$
- $\mathbf{r}^-_{\mathbf{k}} = \mathbf{n}_{C^-T_k} + \mathbf{n}^-_{GT_k}$ et $\mathbf{r}^+_{\mathbf{k}} = \mathbf{n}_{C^+T_k} + \mathbf{n}^+_{GT_k}$
- $\mathbf{n}^-_{GT_k}$ is the normal vector of $[GT_k]$, oriented outward $T_0T_1C^-$, of norm $\|GT_k\|$

As we have introduced the intermediate unknown C_{T_i}, we need an additionnal relation to keep the problem well-posed. It is provided by a least squares interpolation method on $\mathbb{T}_i$, the set of cells incident to the node T_i (cf. Figure 2, right part). For all nodes, C_{T_i} is computed as a linear combination of the unknowns C_j associated with cells in $\mathbb{T}_i$.

A similar formula exists for the three-dimensional case. It can be found in [2], as well as the detailed calculation of (3).

2.3. **Definition of the finite element basis.** On each macroelement K, we construct $(\Phi^i_K)_{1\le i\le n}$ by linear combination of the functions $(\hat{\Psi}^i_{\hat{K}})_{1\le i\le n}$, solutions of (2). To find the coefficients $a_{i,l}$, we impose, for any $1 \le i \le n$, that

$$\forall\, 1 \le j \le n \quad \Phi^i_K(S_j) = \sum_{1\le l\le n} a_{i,l}\Psi^l_{\hat{K}}(S_j) = \delta_{i,j}.$$

Let $\mathcal{I}$ be the set of nodes of the coarse grid, and $N_{\mathcal{I}}$ the cardinal of $\mathcal{I}$. The finite element function Φ^I, associated to $I \in \mathcal{I}$, is defined piecewise on each macroelement $K \ni I$. Provided I coincides with a vertex of K, i.e. $S_j = I$, we have:

$$\Phi^I_{|K} = \Phi^j_K.$$

Due to the oversampling of cells $\hat{K}$, the *Finite Element* basis $(\Phi^I)_{I\in\mathcal{I}}$ is non-conforming, namely each Φ^I is discontinuous through the boundary ∂K of the macroelement K.

A strict oversampling of macroelements is not mandatory to properly define the multiscale method. We can choose $\rho = 0$ and then have $\hat{K} = K$. In such a case, this step of the method becomes trivial as we have $\Phi^i_K = \Psi^i_{\hat{K}}$. However, when $\rho = 0$, the multiscale method is less efficient because of a *boundary layer resonance*, due to the use of linear boundary conditions to solve the cell problems (§2.2). Quantitative and qualitative effects of the oversampling rate ρ are detailed in [14].

2.4. **Solving the coarse problem.** We solve the coarse problem by a *Finite Element Method* relying on the basis $(\Phi^I)_{I\in\mathcal{I}}$. It amounts to solve the linear system

$$\mathbb{K}\mathbf{C}_H = \mathbf{S},$$

with $\mathbb{K}$ a square matrix of size $N_{\mathcal{I}}$, defined by

$$\mathbb{K}(I,J) = \int_\Omega D\nabla\Phi^I \cdot \nabla\Phi^J \quad \forall (I,J) \in \mathcal{I},$$

and $\mathbf{S}$ a column vector deduced from the data f, g and α in (1) with matrices similar to $\mathbb{K}$. We build those matrices by assembling their local counterparts, which allows us to distribute calculus on various processors:

$$\mathbb{K}_K(i,j) = \int_K D\nabla\Phi^i_K \cdot \nabla\Phi^j_K \quad \forall\, 1 \le i,j \le n.$$

We compute these terms, at the fine scale, from the *VFDiam* approximation of fluxes (3), applied to $(\Phi^i_K)_{1\le i\le n}$.

2.5. **Computing the fine-scale solution.** Last but not least, we reconstruct, at the fine scale, the solution C, by weighting the function $(\Phi^I)_{I\in\mathcal{I}}$ with the values of the coarse solution

$$C = \sum_{I\in\mathcal{I}} \mathbf{C}_H(I)\Phi^I \ .$$

As the pratical interest of this method is to solve *very large problems*, where a classical resolution is technically impossible, we usually can not store the reconstructed solution C (or its gradient or its flux density $\|D\nabla C\|$) on the whole domain in a single array. Rather, we reconstruct the solution on subdomains of interest, thus handling arrays of smaller size, fitting in the available computer memory.

This is achieved by gluing solutions of adjacent macroelements, and this is the reason why we need to ensure that meshes are conforming across the boundaries of macroelements (cf. §2.1). However, as the Finite Element method is not conforming, this does not impose any kind of continuity on the solution.

2.6. **Estimate of the computational cost.** We assume that the domain Ω is divided into M macroelements and that each macroelement K is meshed with N microscopic elements. Taking into account the oversampling area, each cell $\hat{K}$ contains about $(1+2\rho)^d N$ elements, with d the space dimension. We note $\mathcal{F}(\mathcal{N})$ a function estimating the cost of solving a linear problem of size $\mathcal{N}$.

The total cost, in computational time, of our multiscale method can be estimated by:

$$\mathcal{C}^{tot} = \underbrace{M \times \mathcal{F}((1+2\rho)^d N)}_{\text{Solving cell problems}} + \underbrace{\mathcal{F}(M)}_{\text{Coarse problem}} + M \times O(N), \tag{4}$$

where we use the Landau notation $O(N)$ for a function of order N at infinity. It is only an approximation of the computational cost of our method, as the costs of some minor operations have been neglected. The rightmost term of (4) represents the cost of assembling Finite Element matrices (§2.4) and of reconstructing the fine-scale solution from the coarse one (§2.5).

Estimate (4) can be compared, for example, to a direct resolution by the Finite Element or Finite Volume method, the cost of which would be $\mathcal{F}(MN)$. Considering that usually $\mathcal{F}(MN) = O(M^3N^3)$ for a direct solver on a full matrix, or $\mathcal{F}(MN) = O(M^2N^2)$ for an iterative solver on a sparse matrix, we can see that the multiscale method is far less expensive than the direct ones. Of course, this low computational cost has a counterpart: results obtained by the multiscale method are, by principle, less accurate.

We would like to emphasize that every step of the multiscale method can be done in parallel but one: solving the coarse problem (§2.4). Therefore, provided we have enough processors, of the order of M, we can expect a very low computational cost on parallel computers:

$$\mathcal{C}^{tot} = \mathcal{F}((1+2\rho)^d N) + \mathcal{F}(M) + O(N). \tag{5}$$

3. Application to a cement paste model.

3.1. **Description of the cement paste.** We consider a cement paste as described by previous CEA works [7]. In this section, lengths are expressed in *micrometers* (μm). The square domain $\Omega = [0, 100]^2$ is composed of a porous phase, of diffusivity $D_0 = 2.2 \times 10^{-9} \mathrm{m}^2 \cdot \mathrm{s}^{-1}$, filled with thousands of inclusions representing the various minerals existing in concrete. Depending on their caracteristics (average size, number of layers and diffusivity), inclusions are dispatched into four groups:

- 3500 homogeneous spheres (diffusivity $d_1 = 1 \times 10^{-20} \mathrm{m}^2 \cdot \mathrm{s}^{-1}$, radius $r_1 \in [0.1, 1.3]$) for the portlandite, aluminates and anhydrides.
- 1700 homogeneous spheres (diffusivity $d_2 = 1 \times 10^{-12} \mathrm{m}^2 \cdot \mathrm{s}^{-1}$, radius $r_2 \in [0.1, 1.3]$) of HD C-S-H (*High Density* C-S-H).
- 1900 homogeneous spheres (diffusivity $d_3 = 9 \times 10^{-12} \mathrm{m}^2 \cdot \mathrm{s}^{-1}$, radius $r_3 \in [0.1, 1.3]$) of LD C-S-H (*Low Density* C-S-H).
- 200 multi-layer spheres (radius $r_4 \in [1.3, 2.8]$) composed of a core of portlandite (stretching on the first 40% of the radius) and two shells: one of HD C-S-H (from $0.4r_4$ to $0.9r_4$) and one of LD C-S-H (from $0.9r_4$).

The random spatial distribution of the inclusions is computed by a program developped by Erwan Adam [3]. Figure 3 presents a close-up of the domain.

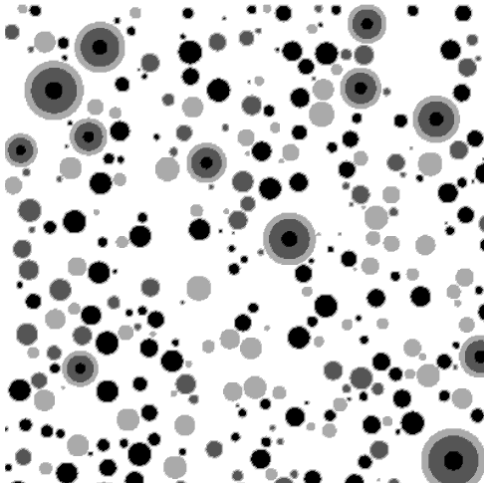


FIGURE 3. Close-up of the cement media: pore phase (white), LD C-S-H (light grey), HD C-S-H (dark grey), portlandite, aluminates and anhydrides (black).

3.2. **Two-dimensional simulation .** We present here our first numerical experiments. They used the *VF9* Finite Volume method, and both the coarse and fine scale meshes are rectangular and structured.

We solved the following problem:

$$\left\{\begin{array}{rcll} -\nabla\cdot(D\,\nabla C) & = & 0 & \text{on } \Omega=[0,100]^2,\\ C(0,y) & = & 1 & \text{for } y\in[0,100],\\ C(1,y) & = & 0 & \text{for } y\in[0,100],\\ D\,\nabla C\cdot n & = & 0 & \text{if } y\in\{0,100\}. \end{array}\right. \tag{6}$$

With these specific conditions, we can compute a physical value of the *homogenized coefficient* from the ingoing and outgoing fluxes, themselves computed from the fine scale solution:

$$D^\star=\frac{1}{L}\int_{\{x=100\}} D\nabla C\cdot n, \tag{7}$$

where $L=100$ is the length of the domain. We emphasize that the homogenized coefficient computed by (7) is *not* the theorical homogenized coefficient as defined by *homogenization theory* [4]. This formula for $D^\star$ has been chosen in order to mimic the experimental measurements of equivalent diffusivity.

The homogenized coefficient $D^\star$ computed by our method is displayed on Figure 4. The number of macroelements K is fixed to 100 as we increase the number of fine scale elements. The coefficient follows a similar evolution for the four values of ρ we tried, matching the figures computed from the direct resolution (solid line) from less that 10%. However, we can not really consider that our homogenized coefficient converges.

This gap is linked to the flux density error, presented on Figure 5. We note that the error is up to 5 times stronger near the boundaries of macroelements K. This *boundary layer resonance*, described in [14], is usually limited by the oversampling method. However this is not the case here. At best, for $\rho=0.05$, we decrease the

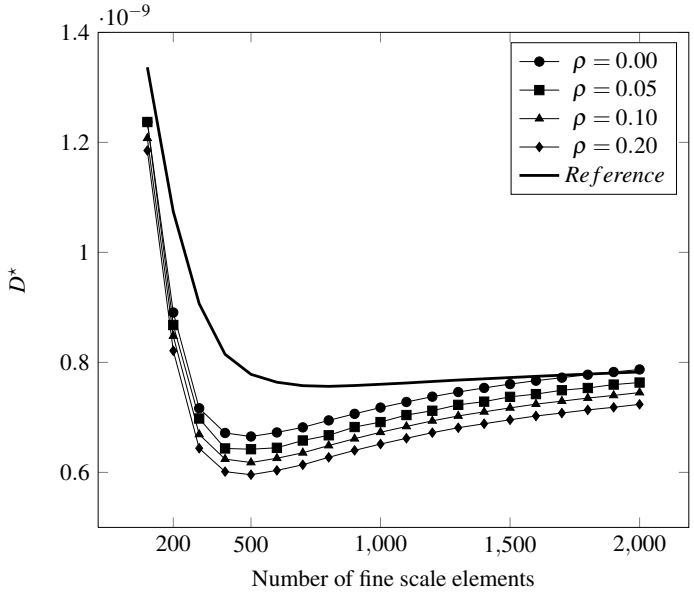


FIGURE 4. Evolution of the homogenized coefficient, with a fixed number M of macroelements, when the precision of the fine-scale mesh increases. The reference solution is computed from a direct resolution of problem (1).

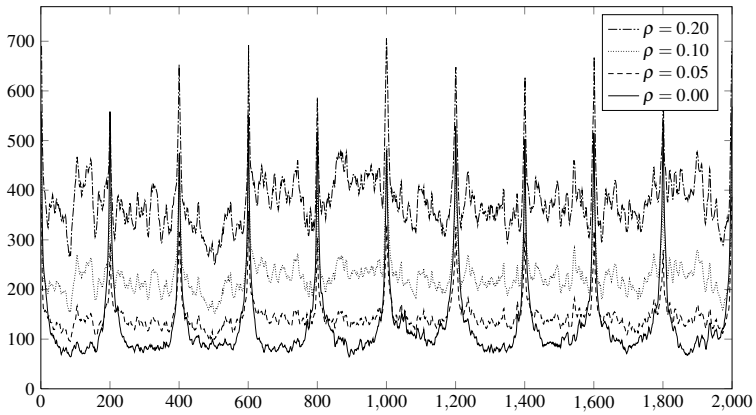


FIGURE 5. Cut across the domain Ω of the flux density: difference between the multiscale method ($M = 10 \times 10$) and the reference solution.

maximal error values by 30%, but we also *increase* the minimal error value. In the end, the mean error value increases.

4. Application to a sample of mortar.

4.1. A $3D$ implementation in a parallel environment.

As shown in section §3, the results computed with our first implementation highlighted some weaknesses

[1]. To improve the method, an important work of implementation has been made, and is presented in this section. Indeed, to increase its performances, the method has been integrated into the parallel numerical code *MPCube* [9], itself relying upon the calculus kernel *Trio-U* [23].

It allows us to easily solve 2D and 3D diffusion problems by the *VFDiam* method, both in sequential and parallel contexts. Our method can now use two levels of parallelism. The *outer-cell* parallelism (see §2.6) allows us to process the work on each cell independently from each other. The *inner-cell* parallelism, using a parallel solver for the computations on each cell, increases our computational capacities to manipulate and solve even larger problems. Applied to the expensive steps of the method, namely meshing the cells and solving the cell problems, these two levels of parallelism broaden significantly our choice of parameters for the multiscale method. For example, we are now able to mesh very accurately each oversampling area and, by extension, to choose the oversampling rate ρ with more precision.

4.2. **Description of the mortar.** We want to conduct simulations on a sample of mortar. In mortars, grains of sand are not directly contiguous to the paste, they are wrapped with highly diffusive *transition layers*. These layers play a prominent part in the transport of chemical species. Indeed, whereas grains of sand are necessarly distinct, the various transition layers can merge with each other. When the density of sand increases, highly diffusive pathes appear through the media, a phenomenon called *percolation* [8].

Experimental processes have estimated the thickness of the transition layers to 30μm, whereas the diameter of grains of sand stretches from 0.16 to 4mm. In order to deal with this wide range in the characteristic lengths, we usually need to work with very fine discretizations, which leads to huge meshes. This is exactly a case where a multiscale method is useful: we can mesh very accurately each part of the domain, with their sand grains and the corresponding layers, but as the domain has been subdivided, each mesh remains of acceptable size.

We consider here a 125mm^3 cubic domain. Like the cement media described in §3, it is modelised as a background media, of adimensionate diffusivity $D_b = 5$, filled with homogeneous spheres, the grains of sand, of diffusivity $D_s = 1$. Each sphere is then wrapped in a 30μm layer of diffusivity $D_t = 15$ acting as a transition zone. We suppose that the transition zones remain distinct from each other, even if it is physically unlikely as we chose the sand volume fraction to be quite high (35%). Diffusivity values come from measures by mercury intrusion porosimetry [8], while the size distribution of sand grains, shown in Figure 6, comes from industrial granulometric measurements [13].

4.3. **Numerical experiments.** The domain is meshed by 125 cubic macroelements (5 in each space direction) with no oversampling ($\rho = 0$). Each cell is then meshed by the tools $GHS3D$ and $BLSURF$ via the $SALOME$ platform [21]. The number of mesh elements in each cell varies from 1×10^5 to 5×10^5, depending on the local geometry of the cell. For example, tangent points between spherical inclusions and the cell boundaries lead to an important increase of the number of tetrahedrons needed to mesh the cell appropriately. This adds up to approximately 2×10^7 elements in the whole domain. Figure 7 presents the mesh for one cell, composed of approximatively 2.5×10^5 elements.

We solve problems which are $3D$ transcriptions of problem (6) used in §3.2. We impose as boundary conditions $C = 1$ when $x = 0$, $C = 0$ when $x = 5$ and a null

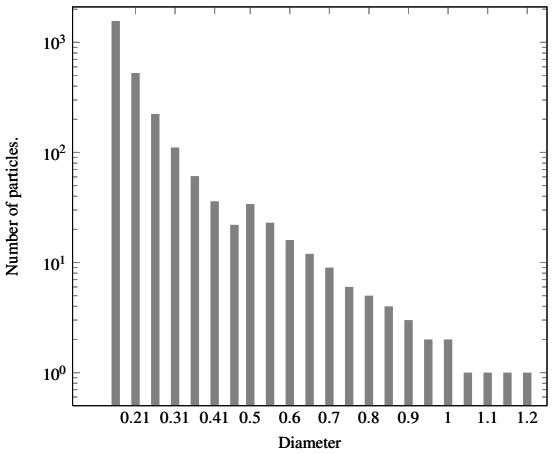


FIGURE 6. Size distribution of sand grains in the mortar, computed from the sand granulometric curves. The biggest sizes of sand grains, from 1.2mm to 4mm, are here dismissed, as the domain is only 5mm wide.

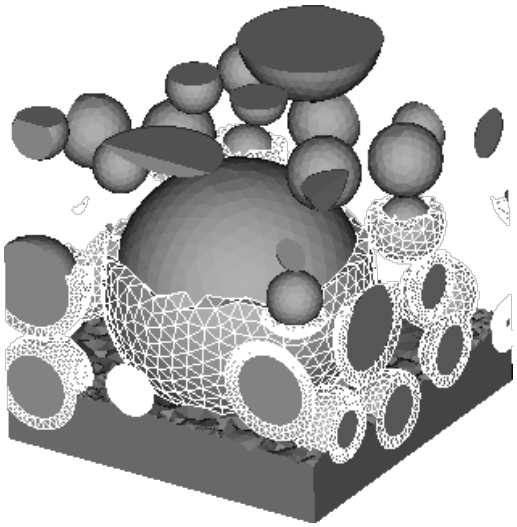


FIGURE 7. Geometry and mesh of one cell. Cement paste (bottom) has been clipped in order to show the spherical sand grains and the matching transition layers (highlighted, clipped on the top).

flux on the other sides. Finally, we choose a null source term. Figure 8 displays the evolution of the concentration C with x, whereas the other coordinates $(y; z)$ are constant. Both global and local phenomena are observed. From a global point of view, the solution C appears almost linear. However, strong variations can occur on very short distances when the chemical species crosses a highly diffusive transition layer.

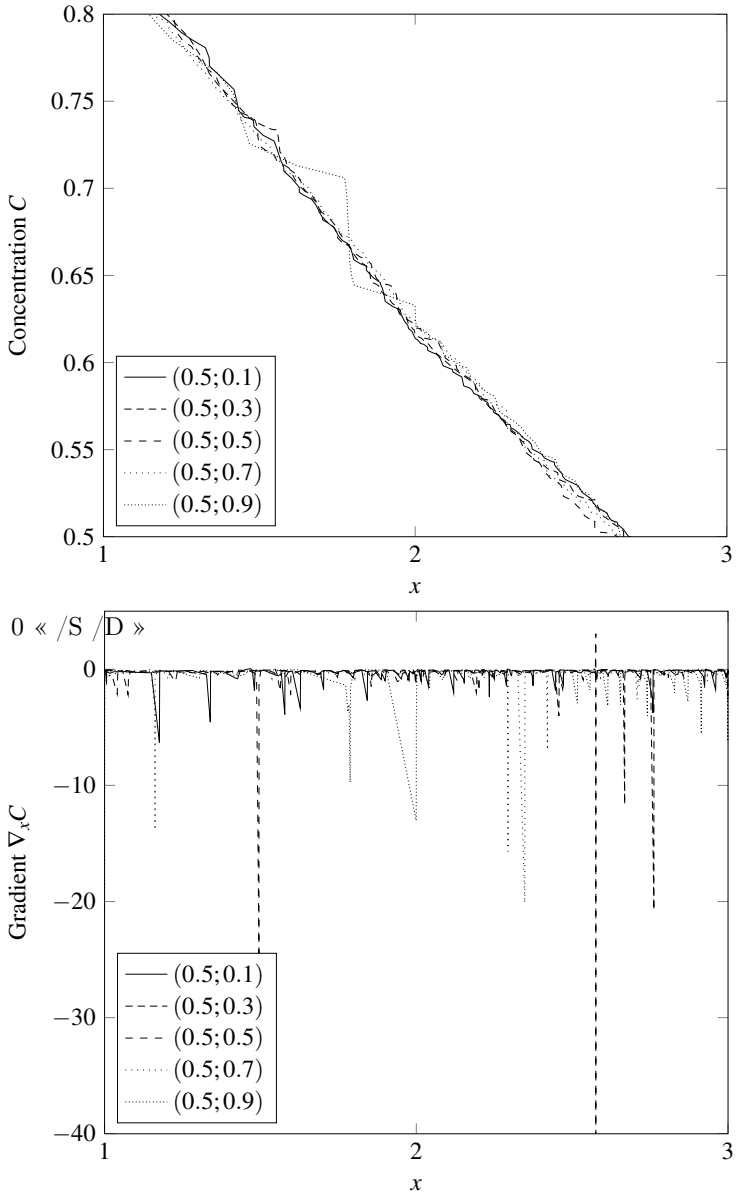


FIGURE 8. Concentration C and longitudinal gradient $\nabla_x C$ across the domain, from $x = 1$ to 3, whereas the remaining coordinates $(y; z)$ are constant. The transition layers are easily recognized by the small jumps they induce in the concentration profiles.

In a future analysis of those results, we will compute the *homogenized coefficient* $D_x^\star$ from the ingoing and outgoing fluxes using (7). By switching the boundary conditions from face to face, we will compute 3 homogenized coefficients $D_x^\star$, $D_y^\star$

and $D_z^\star$. We work here on a representative element of the mortar, which is expected to be isotropic. Consequently, the coefficients should be roughly the same.

5. **Conclusions and future works.** We have presented some numerical experiments using our coupled FV/FE multiscale method. They were first computed with our standalone $2D$ implementation, then with the parallel numerical code *MPCube*.

In the 2D case, our method has shown promising results, but also some weaknesses [1]. Actually, the method converges smoothly for academic benchmarks, but the error tends to stall in the cement media cases because of a boundary layer effect. Our first attempt to solve this problem, by oversampling the coarse elements, has not been successful yet because of the large contrast of diffusivity. To circumvent this difficulty we have emphasized two directions of work.

We have first focused on the oversampling method. It requires a very accurate mesh on each oversampling area and a well-chosen oversampling rate ρ [14]. Using such precision would exceed the computational capacities of our first implementation. Consequently our method was integrated into the parallel numerical code *MPCube*. This new implementation has been tested on a realistic $3D$ case of mortar. More numerical experiments will be conducted, especially comparisons with large size direct resolutions as part of the EHPOC project [15].

Second, it has been made clear that the non-conformity of the Finite Element basis at the coarse scale is the main obstacle for convergence when the oversampling rate ρ is not small. To overcome this problem we plan to use Discontinuous Galerkin methods [10] to solve the coarse problem, instead of a classical Finite Element method.

REFERENCES

[1] T. Abballe, "Une Méthode Multi-échelle Couplée Volumes et Éléments Finis. Application aux Matériaux Cimentaires," Internal CEA Report SFME/LSET/RT/08/003/A, CEA, DEN/DM2S, November 2008.

[2] S. Abide and F. Caro, *Maquette solveur volumes finis multi-physique*, Internal CEA Report SFME/MTMS/RT/07-005/B, CEA, DEN/DM2S, June 2007.

[3] E. Adam, P. Montarnal, B. Bary, H. Peycelon and C. Gallé, *Chaîne de calcul pour la dégradation sous eau des bétons: méthodologie, outil et illustration*, Internal CEA Report SFME/MTMS/RT/07-0006/A, CEA, DEN/DM2S, May 2007.

[4] G. Allaire, "Shape Optimization by the Homogenization Method," Applied Mathematical Sciences, vol. **146**, Springer, 2001.

[5] G. Allaire and R. Brizzi, *A multiscale finite element method for numerical homogenization*, Multiscale Modeling and Simulation, **4** (2005), 790–812.

[6] S. Bejaoui and B. Bary, *Modeling of the link between microstructure and effective diffusivity of cement pastes using a simplified homogenization method*, Cement and Concrete Research, **37** (2007), 469–480.

[7] S. Bejaoui, B. Bary and C. Julien, *Modélisation de la relation entre microstructure et diffusivité effective des pâtes de ciment à partir d'une méthode numérique 3d*, Internal CEA Report DPC/SCCME/05-327-A, CEA, Département de Physico-Chimie, January 2006.

[8] B. Bourdette, E. Ringot and J. P. Ollivier, *Modelling of the transition zone porosity*, Cement and Concrete Research, **25** (1995), 741–751.

[9] F. Caro and E. Laucoin, "Description du Module MPCube Permettant la Simulation Numérique des Écoulements Diphasiques à deux Composants en Milieu Poreux," Internal CEA Report SFME/LSET/RT/09-006/A, CEA, DEN/DM2S, April 2009.

[10] B. Cockburn, *Discontinuous galerkin methods*, ZAMM - Journal of Applied Mathematics and Mechanics, **83** (2003), 731–754.

[11] Y. Coudière, J-P. Vila and P. Villedieu, *Convergence rate of a finite volume scheme for a two dimensional convection-diffusion problem*, Mathematical Modelling and Numerical Analysis, **33** (1999), 493–516.

[12] W. E and B. Engquist, *The heterogeneous multiscale methods*, Communications in Mathematical Sciences, **1** (2003), 87–132.
[13] T. Eder, H. Rötzer, D. Donhoffer, L. Riedlmayer, W. Volkmann, K. Wallisch and J. Zöhrer, *Online measurement of the grain size distribution of sand 0-4 mm used for the production of concrete*, Particle and Particle Systems Characterization, **1** (1984), 85–88.
[14] Y. R. Efendief, T. Y. Hou and X-H. Wu, *Convergence of a nonconforming multiscale finite element method*, SIAM Journal of Numerical Analysis, **37** (2000), 888–910.
[15] *Environnement Haute Performance pour l'Optimisation et la Conception*, http://www.systematic-paris-region.org/fr/UserFiles/File/EHPOC.pdf.
[16] I. Faille, *A control volume method to solve an elliptic equation on a two-dimensional irregular mesh*, Comput. Methods Appl. Mech. Eng., **100** (1992), 275–290.
[17] H. Hajibeygi, G. Bonfigli, M. A. Hesse and P. Jenny, *Iterative multiscale finite-volume method*, Journal of Computational Physics, **227** (2008), 8604–8621.
[18] T. Y. Hou and X-H. Wu, *A multiscale finite element method for elliptic problems in composite materials and porous media*, Journal of Computational Physics, **134** (1997), 169–189.
[19] P. Jenny, S. H. Lee and H. A. Tchelepi, *Multiscale finite-volume method for elliptic problems in subsurface flow simulation*, Journal of Computational Physics, **187** (2003), 47–67.
[20] A. M. Matache, I. Babuška and C. Schwab, *Generalized* p*-FEM in homogenization*, Numerische Mathematik, **86** (2000), 319–375.
[21] *Plate-forme SALOME*, http://www.salome-platform.org/.
[22] C. Schwab and A. M. Matache, *Multiscale and multiresolution methods: Theory and applications*, Lecture Notes in Computational Science and Engineering, vol. **20**, ch. Generalized FEM for Homogenization Problems, 197–238, Springer Verlag, 2002.
[23] *Trio-U*, http://www-trio-u.cea.fr/.

Received January 2010; revised June 2010.

E-mail address: thomas.abballe@cea.fr
E-mail address: gregoire.allaire@polytechnique.edu
E-mail address: philippe.montarnal@cea.fr
E-mail address: eli.laucoin@cea.fr

Table des figures

1.1 Structure apparente du béton à différentes échelles d'observation. . 17

2.1 Discrétisations globale $T_H(\Omega)$ et locale $T_h(\hat{K})$ du domaine Ω. . . . 30
2.2 Illustration des méthodes de Galerkin directes, de Galerkin approchées et de Volumes Finis multi-échelles sur un maillage grossier. 36
2.3 Construction du maillage dual $\tilde{T}_H(\Omega)$ à partir du maillage primal $T_H(\Omega)$. 39

3.1 Division du domaine Ω en macroélément K et construction des cellules $\hat{K}$ associées. 46
3.2 Points et droites introduits par le schéma *VF9*. 49
3.3 Schéma d'un élément *VFDiam* et de l'ensemble $\mathbb{S}_0$. 51
3.4 Fonction de base de la méthode d'Éléments Finis. 55
3.5 Fonction de base de la méthode Discontinue de Galerkin. 60
3.6 Illustration du phénomène de percolation dans le béton. 67

4.1 Organigramme simplifié de l'implémentation de la méthode Q_1/*VF9*. 75
4.2 Exemple 4.2 : représentation de $e_2(u_h)$ en fonction de h. 77
4.3 Exemple 4.3 : représentation de $e_2(u_h)$ en fonction de h. 78
4.4 Exemple 4.4 : erreur sur les coefficients de $\mathbb{K}_K$ en fonction de h. . 80
4.5 Exemple 4.4 : erreur sur les coefficients de $\mathbb{M}_K$ en fonction de h. . 80
4.6 Exemple 4.4 : erreur sur les coefficients de $\mathbb{M}_{b,F}$ en fonction de h. 81
4.7 Exemple 4.5 : erreur sur les coefficients matriciels en fonction de h. 82
4.8 Exemples 4.6, 4.7 et 4.8 : erreur $e_\infty(C_H)$ en fonction de H. 84
4.9 Exemple 4.9 : diffusivité et solution de référence. 86
4.10 Exemple 4.9 : erreurs $e_2(C_{H,h})$ et $e_2(D\nabla C_{H,h})$ à H fixé. 87
4.11 Exemple 4.9 : erreurs $e_2(C_{H,h})$ et $e_2(D\nabla C_{H,h})$ à h/H fixé. 88
4.12 Attribution de la diffusivité sur un maillage régulier. 90
4.13 Intersections d'un cercle C et d'un rectangle R 92
4.14 Aires de surfaces élémentaires. 94
4.15 Représentation de la pâte cimentaire modélisée. 95
4.16 Exemple 4.10 : $e_2(C_{H,h})$, $e_2(\nabla C_{H,h})$ et $e_2(D\nabla C_{H,h})$ à H fixé. . . . 97
4.17 Exemple 4.10 : évolution de la diffusivité équivalente D^* à H fixé. 98
4.18 Exemple 4.10 : évolution de la diffusivité équivalente D^* à h/H fixé. 98

4.19 Exemple 4.10 : erreurs $e_2(C_{H,h})$, $e_2(\nabla C_{H,h})$ et $e_2(D\nabla C_{H,h})$ à h/H fixé. 99
4.20 Coupe de la densité de flux $D_h \nabla C_{H,h} \cdot \mathbf{n}$ à travers le domaine Ω. . . 100

5.1 Organigramme de la chaîne de calcul *SALOME-MPCube*. 107
5.2 Liste des fichiers nécessaires à une simulation *MPCube*. 109
5.3 Un fichier *data* lançant trois instances parallèles de *MPCube*. . . . 110
5.4 Commandes *MPCube* pour la construction de la base locale sur une cellule. 111
5.5 Commandes *MPCube* : passage du format LATA au format MED. 113
5.6 Commandes *MPCube* : mise en forme parallèle des résultats. . . . 114
5.7 Commandes *MPCube* : résolution du problème grossier. 115
5.8 Exemple 5.1 : erreur sur les coefficients de $\mathbb{K}_K$ en fonction de h. . 124
5.9 Exemple 5.1 : erreur sur les coefficients de $\mathbb{M}_K$ en fonction de h. . 124
5.10 Exemple 5.1 : erreur sur les coefficients de $\mathbb{M}_{b,F}$ en fonction de h. 124
5.11 Exemple 5.2 : erreur sur les coefficients matriciels en fonction de h. 125
5.12 Exemples 5.3, 5.4 et 5.5 : erreur $e_\infty(C_H)$ en fonction de H. 129
5.13 Exemples 5.6, 5.7 et 5.8 : erreur $e_\infty(C_H)$ en fonction de H. 130

6.1 Étapes de la génération du maillage $T_h(\hat{K})$ par l'Algorithme 6.1. . 139
6.2 Répartition du temps de génération des maillages $T_h(\Omega_n)$. 143
6.3 Captures d'écran de *SALOME* lors du travail de l'Algorithme 6.1 sur l'Exemple 6.1 144
6.4 Assemblage de la géométrie d'une cellule par l'Algorithme 6.3. . 146
6.5 Intersection et coupe de deux ensembles A et B. 148
6.6 Ajout de points à une face par l'Algorithme 6.6 154
6.7 Temps de génération du milieu Ω_n pour les Algorithmes 6.1 et 6.7. 155
6.8 Maillages coïncidents sur la frontière commune de deux macroéléments voisins. 156
6.9 Maillages coïncidents par jeux successifs : principe. 158
6.10 Maillages coïncidents par jeux successifs : le problème des inclusions. 159
6.11 Maillages coïncidents par maillage de peau : vues *SALOME* . . . 161
6.12 Adaptation de la discrétisation d'un milieu au cours de sa dégradation. 167

7.1 Exemples 7.1.1, 7.1.2 et 7.1.3 : erreur $e_\infty(C_H)$ en fonction de H. . 172
7.2 Exemple 7.2 : fonction Φ_K^0 issue de la cellule $\hat{K} = (0,0)$. 174
7.3 Exemple 7.2 : erreur $e_2(C_{H,h})$ en fonction de H, à ρ fixé. 176
7.4 Exemple 7.2 : erreur $e_2(C_{H,h})$ en fonction de H, à α fixé. 177
7.5 Exemple 7.2 : écart entre C et $C_{H,h}$. 178
7.6 Exemple 7.3 : temps de génération des maillages fins $T_h(\hat{K})$. . . 180
7.7 Exemple 7.3 : temps de résolution des problèmes locaux. 181
7.8 Exemple 7.4 : distribution de la taille des grains de sable. 183
7.9 Exemple 7.4 : maillage d'une cellule du domaine. 184
7.10 Exemple 7.4.1 : concentration $C_{H,h}^x$ et gradient longitudinal $\nabla_x C_{H,h}^x$. 186

7.11 Exemple 7.5 : distribution de la taille des inclusions. 189
7.12 Exemple 7.5 : maillage du macroélément K. 190
7.13 Exemple 7.6 : distribution de la taille des inclusions. 195
7.14 Exemple 7.6 : maillages d'une cellule $\hat{K}$ pour $\rho = 0$ et $\rho = 0.2$. . . 196
7.15 Exemple 7.6 : temps de résolution des problèmes locaux. 197
7.16 Exemple 7.6.1 : visualisation de la solution fine $C_{H,h}$. 198

A.1 Illustration d'un élément *VFDiam* en dimension 3. 213

Table des Algorithmes.

2.1 Algorithme d'une méthode de simulation multi-échelle. 30
3.1 Étapes des méthodes multi-échelles. 44
3.2 Application des conditions aux limites de Dirichlet. 58
4.1 Aire d'intersection entre un cercle et un rectangle. 93
5.1 Objet BASE_LOCALE : construction de la base locale $(\Phi_K^l)_{1\leq l\leq n}$. . 119
5.2 Objet BASE_LOCALE : calcul des matrices locales. 121
5.3 Objet PROBLEME_GROSSIER : résolution du problème grossier. . 126
6.1 Génération des maillages de cellules $T_h(\hat{K})$. 138
6.2 Répartition des inclusions $\mathscr{I}$ par cellule. 140
6.3 Assemblage de la géométrie d'une cellule. 145
6.4 Intersection d'un grand nombre d'objets avec un unique objet. . . 149
6.5 Coupe d'un objet de référence par un grand nombre d'objets. . . . 150
6.6 Ajout d'un ensemble de points à un rectangle. 153
6.7 Génération des maillages de cellules $T_h(\hat{K})$. 154
6.8 Génération du maillage $2D$ de référence. 162
6.9 Génération des maillages de cellules $T_h(\hat{K})$ coincidents. 163
6.10 Construction de Ω_h^i à partir de Ω_h^{i+1}. 165

Bibliographie

[1] AARNES J.E. On the use of a mixed multiscale finite element method for greater flexibility and increased speed or improved accuracy in reservoir simulation. *Multiscale Modeling & Simulation*, 2(3), pages 421–439. doi : 10.1137/030600655. 2004.

[2] AARNES J.E., EFENDIEV Y. et JIANG L. Mixed multiscale finite element methods using limited global information. *Multiscale Modeling & Simulation*, 7(2), pages 655–676. doi :10.1137/070688481. 2008.

[3] AARNES J.E. et HEIMSUND B.O. Multiscale discontinuous Galerkin methods for elliptic problems with multiple scales. Multiscale Methods in Science and Engineering, *Lecture Notes in Computational Science and Engineering*, tome 44, pages 1–20. Springer Berlin Heidelberg. doi : 10.1007/3-540-26444-2_1. 2005.

[4] AARNES J.E., KIPPE V. et LIE K.A. Mixed multiscale finite elements and streamline methods for reservoir simulation of large geomodels. *Advances in Water Resources*, 28(3), pages 257–271. doi :10.1016/j.advwatres.2004.10.007. 2005.

[5] AAVATSMARK I. Comparison of monotonicity for some multipoint flux approximation methods. R. Eymard et J.M. Hérard (rédacteurs), Finite Volumes for Complex Applications, tome 5, pages 19 – 34. 2008.

[6] Abaqus FEA. `http://www.simulia.com/products/abaqus_fea.html`.

[7] ABBALLE T. Méthode multi-échelle VF/EF pour la modélisation des matériaux cimentaires. Journées Scientifiques du GdR MoMaS, Fréjus. Poster. 2007.

[8] ABBALLE T. A coupled FV/FE multiscale method applied to cement media. 20ème séminaire sur la mécanique des fluides numérique, Paris. Poster. 2008.

[9] ABBALLE T. Une méthode multi-échelle couplée Volumes et Éléments finis. Application aux matériaux cimentaires. Rapport technique SFME/LSET/RT/08/003/A, CEA, DEN/DM2S. 2008.

[10] ABBALLE T. Application of a coupled FV/FE multiscale method to cement media. SIAM Conference on Mathematical and Computational Issues in the Geosciences, Leipzig. Communication orale. 2009.

[11] ABBALLE T. Application of a coupled FV/FE multiscale method to cement media. Congrès New Trends in Model Coupling, Paris. Poster. 2009.

[12] ABBALLE T. Une méthode multi-échelle couplée VF/EF appliquée aux matériaux cimentaires. Congrès SMAI 2009, 4e Biennale Française des Mathématiques Appliquées et Industrielles. Communication orale. 2009.

[13] ABBALLE T. Simulation multi-échelle et homogénéisation des matériaux cimentaires. 10ème Journées scientifiques de la DEN/DANS. Communication orale. 2010.

[14] ABBALLE T., ALLAIRE G., LAUCOIN E. et MONTARNAL P. Application of a coupled FV/FE multiscale method to cement media. *Networks and Heterogeneous Media*, 5(3), pages 603–615. doi :10.3934/nhm.2010.5.603. 2010.

[15] ABDULLE A. On a priori error analysis of fully discrete heterogeneous multiscale FEM. *Multiscale Modeling & Simulation*, 4(2), pages 447–459. doi :10.1137/040607137. 2005.

[16] ABDULLE A. Multiscale method based on discontinuous galerkin methods for homogenization problems. *Comptes Rendus de l'Académie des Sciences de Paris*, 346(1-2), pages 97–102. doi :10.1016/j.crma.2007.11.029. 2008.

[17] ABDULLE A. et NONNENMACHER A. A posteriori error analysis of the heterogeneous multiscale method for homogenization problems. *Comptes Rendus Mathematique*, 347(17-18), pages 1081–1086. doi :10.1016/j.crma.2009.07.004. 2009.

[18] ABDULLE A. et SCHWAB C. Heterogeneous multiscale FEM for diffusion problems on rough surfaces. *Multiscale Modeling & Simulation*, 3(1), pages 195–220. doi :10.1137/030600771. 2005.

[19] ABIDE S. Développement d'un module de convection/diffusion en milieu poreux à l'aide du noyau parallèle trio-u : spécifications et conceptions. Rapport technique CS/311-1/AB06A043/RAP/06/076, CEA, DEN/DANS/DM2S/SFME/MTMS. 2007.

[20] ABIDE S. et CARO F. Maquette solveur volumes finis multi-physique. Rapport technique SFME/MTMS/RT/07-005/B, CEA, DEN/DM2S. 2007.

[21] ADAM E., MONTARNAL P., BARY B., PEYCELON H. et GALLÉ C. Chaîne de calcul pour la dégradation sous eau des bétons : méthodologie, outil et illustration. Rapport technique SFME/MTMS/RT/07-0006/A, CEA Saclay DEN/DM2S. 2007.

[22] ADAM J.P. La construction romaine : matériaux et techniques. Grands manuels Picard. Picard. 1984.

[23] AGENCE NATIONALE POUR LA GESTION DES DÉCHETS RADIOACTIFS (rédacteur). Argile - Synthèse : Évaluation de la faisabilité du stockage géologique en formation argileuse. Numéro 266B dans Les Rapports. Andra. 2005.

[24] AGENCE NATIONALE POUR LA GESTION DES DÉCHETS RADIOACTIFS (rédacteur). Recherches de l'Andra sur le stockage géologique des déchets radioactifs à haute activité et à vie longue : Résultats et perspectives. Numéro 265 dans Les Rapports. Andra. 2005.

[25] AGENCE NATIONALE POUR LA GESTION DES DÉCHETS RADIOACTIFS (rédacteur). Matières et déchets radioactifs. Loi de programme du 28 juin 2006 : texte consolidé par l'Andra. Numéro 305C dans Les Dossiers. Andra. 2009.

[26] ALLAIRE G. Homogenization and two-scale convergence. *SIAM Journal on Mathematical Analysis*, 23(6), pages 1482–1518. doi :10.1137/0523084. 1992.

[27] ALLAIRE G. Shape optimization by the homogenization method, *Applied Mathematical Sciences*, tome 146. Springer. 2001.

[28] ALLAIRE G. et BRIZZI R. A multiscale finite element method for numerical homogenization. *Multiscale Modeling and Simulation*, 4(3), pages 790–812. doi :10.1137/040611239. 2005.

[29] ARNOLD D.N. An interior penalty finite element method with discontinuous elements. *SIAM Journal of Numerical Analysis*, 19(4), pages 742–760. 1982.

[30] ARNOLD D.N., BREZZI F., COCKBURN B. et MARINI L.D. Unified analysis of discontinuous galerkin methods for elliptic problems. *SIAM Journal on Numerical Analysis*, 39(5), pages 1749–1779. doi :10.1137/S0036142901384162. 2002.

[31] BABUŠKA I. et ZLÁMAL M. Nonconforming elements in the finite element method with penalty. *SIAM Journal on Numerical Analysis*, 10(5), pages 863–875. 1973.

[32] BABUŠKA I., BANERJEE U. et OSBORN J.E. Survey of meshless and generalized finite element methods : A unified approach, *Acta Numerica*,

tome 12. Cambridge University Press. doi :10.1017/S0962492902000090. 2003.

[33] BABUŠKA I., CALOZ G. et OSBORN J. Special finite element methods for a class of second order elliptic problems with rough coefficients. *SIAM Journal on Numerical Analysis*, 31(4), pages 945–981. doi :10.1137/0731051. 1994.

[34] BAKHVALOV N. et PANASENKO G. Homogenization : Averaging Processes in Periodic Media, *Mathematics and its Applications*, tome 36. Kluwer Academic Publishers. 1989.

[35] BARNES J. et HUT P. A hierarchical $O(N \log N)$ force-calculation algorithm. *Nature*, 324, pages 446 – 49. doi :10.1038/324446a0. 1986.

[36] BARY B., BEN-HAHA M., ADAM E. et MONTARNAL P. Numerical and analytical effective elastic properties of degraded cement pastes. *Cement and Concrete Research*, 39(10), pages 902–912. doi :10.1016/j.cemconres.2009.06.012. 2009.

[37] BASSI F. et REBAY S. A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier-Stokes equations* 1. *Journal of Computational Physics*, 131(2), pages 267–279. 1997.

[38] BASSI F., REBAY S., MARIOTTI G., PEDINOTTI S. et SAVINI M. A high-order accurate discontinuous finite element method for inviscid and viscous turbomachinery flows. 2nd European Conference on Turbomachinery Fluid Dynamics and Thermodynamics (Antwerpen, Belgium)(R. Decuypere and G. Dibelius, eds.), Technologisch Instituut, pages 99–108. 1997.

[39] BAUMANN C. et ODEN J. A discontinuous hp finite element method for convection–diffusion problems. *Computer Methods in Applied Mechanics and Engineering*, 175(3-4), pages 311–341. 1999.

[40] BECCANTINI A. et GOUNAND S. Evaluation of the diffusive terms of the Navier-Stokes equation via cell-centered finite volume approach. Rapport technique SFME/LTMF/RT/02-024/A, CEA DEN. 2002.

[41] BEJAOUI S. Modélisation du transport diffusif des bétons. synthèse bibliographique et analyse comparative de méthodes d'homogénéisation. version 2.0 du modèle d'homogénéisation simplifié microstructure/transport MICROTRANS. Rapport technique DPC/SCCME/03-222-A, CEA, Département de Physico-Chimie. 2003.

[42] BEJAOUI S. et BARY B. Modeling of the link between microstructure and effective diffusivity of cement pastes using a simplified composite model. *Cement and Concrete Research*, 37(3), pages 469–480. doi : 10.1016/j.cemconres.2006.06.004. 2007.

[43] BEJAOUI S., BARY B. et JULIEN C. Modélisation de la relation entre microstructure et diffusivité effective des pâtes de ciment à partir d'une méthode numérique 3d. Rapport technique DPC/SCCME/05-327-A, CEA, Département de Physico-Chimie. 2006.

[44] BENGAOUER A., CHAVANT C., LOTH L. et MONTARNAL P. ALLIANCES : Simulation platform for nuclear waste storages and disposal. *European Nuclear Features*, 4, pages 13–14. 2005.

[45] BENGAOUER A., MONTARNAL P., PEYCELON H., GALLE C. et BARY B. Projet IOLS (Infrastructure et Outils Logiciel pour la Simulation. Rapport technique SFME/MTMS/RT/06/018/A, CEA, DEN/DM2S. 2006.

[46] BENSOUSSAN A., LIONS J. et PAPANICOLAOU G. Asymptotic analysis for periodic structures. North Holland. 1978.

[47] BERNARD-MICHEL G., LE POTIER C., BECCANTINI A., GOUNAND S. et CHRAIBI M. The Andra Couplex 1 test case : Comparisons between finite-element, mixed hybrid finite element and finite volume element discretizations. *Computational Geosciences*, 8(2), pages 187 – 201. doi : 10.1023/B:COMG.0000035079.68284.49. 2004.

[48] BLUM J., CARTALADE A., CAVANNA B. et MONTARNAL P. Estimation des paramètres de transport d'un milieu poreux, approche par état adjoint. Rapport technique RT/02-018/A, CEA, DEN/DM2S. 2002.

[49] BLUM J., CARTALADE A., CAVANNA B. et MONTARNAL P. Paramétrisation automatique des coeffficients de transport d'un milieu poreux, approche par état adjoint. Rapport technique RT/03-002/A, CEA, DEN/DM2S. 2003.

[50] BONNET-BENDHIA A.S., LUNÉVILLE E. et HAZARD C. Résolution numérique des équations aux dérivées partielles. ENSTA, Cours MA201. 2005.

[51] BOURBAKI N. Topologie générale : chapitres 1 à 4, *Éléments de mathématique*, tome 1. Hermann. 1971.

[52] BOURDETTE B., RINGOT E. et OLLIVIER J. Modelling of the transition zone porosity. *Cement and concrete research*, 25(4), pages 741–751. doi : 10.1016/0008-8846(95)00064-J. 1995.

[53] BOURGAT J.F. et DERVIEUX A. Méthode d'homogénéisation des opérateurs périodiques : étude des correcteurs provenant du développement asymptotique. Rapport technique 278, IRIA Laboratoire de Recherche en Informatique et Automatique. 1978.

[54] BREZIS H. Analyse fonctionnelle : Théorie et applications. Sciences Sup. Dunod. 2005.

[55] BREZZI F. et FORTIN M. Mixed and hybrid finite element methods, *Springer Series in Computational Mathematics*, tome 15. Springer-Verlag. 1991.

[56] BREZZI F., MANZINI G., MARINI D., PIETRA P. et RUSSO A. Discontinuous finite elements for diffusion problems. *Atti Convegno in onore di F. Brioschi (Milano 1997), Istituto Lombardo, Accademia di Scienze e Lettere*, pages 197–217. 1999.

[57] BREZZI F., MANZINI G., MARINI D., PIETRA P. et RUSSO A. Discontinuous Galerkin approximations for elliptic problems. *Numerical Methods for Partial Differential Equations*, 16(4), pages 365–378. 2000.

[58] BUFFA A., HUGHES T.J.R. et SANGALLI G. Analysis of a multiscale discontinuous galerkin method for convection-diffusion problems. *SIAM Journal on Numerical Analysis*, 44(4), pages 1420–1440. doi :10.1137/050640382. 2006.

[59] CAMPBELL D. et FOLK R. The ancient Egyptian pyramids – concrete or rock. *Concrete International*, 13(8), pages 28–30. 1991.

[60] CARO F. Évaluation et adaptation des solveurs pour la diffusion dans des milieux hétérogènes pour de grands volumes de données, livrable EHPOC L 4.1.1. Rapport technique LSET/RT/10-013/A, CEA, DEN/DANS. 2010.

[61] CARO F. et LAUCOIN E. Description du module MPCube permettant la simulation numérique des écoulements diphasiques à deux composants en milieu poreux. Rapport technique SFME/LSET/RT/09-006/A, CEA, DEN/DM2S. 2009.

[62] CARO F. et LAUCOIN E. Description du module MPCube permettant la simulation numérique des écoulements diphasiques à deux composants en milieu poreux. Rapport technique SFME/LSET/RT/09-006/A, CEA, DEN/DM2S. 2009.

[63] CARTALADE A., LATRILLE C., LAPASSET G., CHAMBELLAN D. et CADALEN S. Dispositif expérimental BEETI : démarche d'identification d'une loi de dispersivité, interprétation des mesures dichromatiques X et conception d'un système d'étalonnage. Rapport technique SFME/MTMS/RT/07-018/A, CEA, DEN/DM2S. 2007.

[64] CASTILLO P. Performance of discontinuous galerkin methods for elliptic pdes. *SIAM Journal on Scientific Computing*, 24(2), pages 524–547. doi : 10.1137/S1064827501388339. 2002.

[65] CHAN T.F. et VAN DER VORST H.A. Approximate and incomplete factorizations. S.A. Keyes D et V. Venkatakrishnan (rédacteurs), Parallel Numerical Algorithms, *ICASE/LaRC Interdisciplinary Series in Science and Engineering*, tome 11, pages 167 – 202. 1997.

[66] CHAPTAL J.A. Observations sur quelques avantages qu'on peut retirer des terres ocreuses, avec les moyens de les convertir en brun rouge, et d'en former des pozzolanes propres à remplacer avec économie les étrangères et les nationales. Impression des États de Languedoc. 1787.

[67] CHEN Z. Multiscale methods for elliptic homogenization problems. *Numerical Methods for Partial Differential Equations*, 22(2), pages 317–360. doi :10.1002/num.20099. 2006.

[68] CHEN Z., CUI M., SAVCHUK T.Y. et YU X. The multiscale finite element method with nonconforming elements for elliptic homogenization problems. *Multiscale Modeling & Simulation*, 7(2), pages 517–538. doi : 10.1137/070691917. 2008.

[69] CHEN Z. et HOU T.Y. A mixed multiscale finite element method for elliptic problems with oscillating coefficients. *Mathematics of Computation*, 72(242), pages 541–576. doi :10.1090/S0025-5718-02-01441-2. 2003.

[70] CHOMAT L., LAUCOIN E., ABBALLE T., ADAM E., BARY B., MONTARNAL P. et GALLÉ C. Chaîne de calcul pour la dégradation sous eau des bétons (ChaCalBe) : Application du démonstrateur au calcul de coefficients de diffusion pour un VER matériau cimentaire et validation sous SALOME. Rapport technique SFME/LSET/RT/10/011/A, CEA Saclay DEN/DM2S. 2010.

[71] CIARLET P. The finite element method for elliptic problems. North Holland. 1978.

[72] CIORANESCU D. et DONATO P. An introduction to Homogenization. Oxford University Press. 1999.

[73] CLÉMENT F., KHVOENKOVA N., CARTALADE A. et MONTARNAL P. Analyse de sensibilité et estimation de paramètres de transport pour une équation de diffusion, approche par état adjoint. Rapport technique 5132, Inria. 2004.

[74] COCKBURN B. Discontinuous galerkin methods. *ZAMM - Journal of Applied Mathematics and Mechanics*, 83(11), pages 731–754. doi :10.1002/zamm.200310088. 2003.

[75] COCKBURN B. et SHU C. The local discontinuous Galerkin method for time-dependent convection-diffusion systems. *SIAM Journal on Numerical Analysis*, 35(6), pages 2440–2463. 1998.

[76] Cast3M 2010. `http://www-cast3m.cea.fr`.

[77] Code_Aster. `http://www.code-aster.org`.

[78] CONSTANTINIDES G. et ULM F.J. The effect of two types of C-S-H on the elasticity of cement-based materials : Results from nanoindentation and micromechanical modeling. *Cement and Concrete Research*, 34(1), pages 67–80. doi :10.1016/S0008-8846(03)00230-8. 2004.

[79] COUDIÈRE Y., VILA J.P. et VILLEDIEU P. Convergence rate of a finite volume scheme for a two dimensional convection-diffusion problem. *Mathematical Modelling and Numerical Analysis*, 33(3), pages 493–516. doi : 10.1051/m2an:1999149. 1999.

[80] Langage de programmation C++. `http://www.open-std.org/jtc1/sc22/wg21/`.

[81] DE MARSILY G. Quantitative Hydreogeology : Groundwater Hydrology for Engineers. Academic Press. 1986.

[82] DOUGLAS J. et DUPONT T. Interior penalty procedures for elliptic and parabolic Galerkin methods. R. Glowinski et J. Lions (rédacteurs), Computing Methods in Applied Sciences, *Lecture Notes in Physics*, tome 58, pages 207–216. Springer. doi :10.1007/BFb0120591. 1976.

[83] DRIDI W. Modélisation du transport diffusif dans les bétons. synthèse bibliographique et analyse des besoins pour l'évolution de la modélisation "Microtrans" aux milieux insaturés. Rapport technique SCCME/RT/07-739/A, CEA Saclay DEN/DPC. 2008.

[84] DRIDI W. et MEDDAHI N. Modélisation du transport diffusif dans les bétons. synthèse des travaux de modélisation par l'approche "Microtrans" en milieu saturé. Rapport technique SCCME/RT/07-762/A, CEA Saclay DEN/DPC. 2008.

[85] DRYJA M. A Neumann-Neumann algorithm for a mortar discretization of elliptic problems with discontinuous coefficients. *Numerische Mathematik*, 99, pages 645–656. doi :10.1007/s00211-004-0573-2. 2005.

[86] E W. et ENGQUIST B. The heterogeneous multiscale methods. *Communications in Mathematical Sciences*, 1(1), pages 87–132. MR1979846. 2003.

[87] E W., MING P. et ZHANG P. Analysis of the heterogeneous multiscale method for elliptic homogenization problems. *Journal of the American Mathematical Society*, 18(1), pages 121–156. doi :10.1090/S0894-0347-04-00469-2. 2005.

[88] ECKARDT S. et KÖNKE C. Adaptive damage simulation of concrete using heterogeneous multiscale models. *Journal of Algorithms & Computational Technology*, 2(2), pages 275–297. doi :10.1260/174830108784646661. 2008.

[89] EDER T., RÖTZER H., DONHOFFER D., RIEDLMAYER L., VOLKMANN W., WALLISCH K. et ZÖHRER J. Online measurement of the grain size distribution of sand 0-4 mm used for the production of concrete. *Particle and Particle Systems Characterization*, 1(1-4), pages 85–88. doi :10.1002/ppsc.19840010114. 1984.

[90] EFENDIEF Y.R., HOU T.Y. et WU X.H. Convergence of a nonconforming multiscale finite element method. *SIAM Journal of Numerical Analysis*, 37(3), pages 888–910. doi :10.1137/S0036142997330329. 2000.

[91] EFENDIEV Y., HOU T. et GINTING V. Multiscale finite element methods for nonlinear problems and their applications. *Communications in Mathematical Sciences*, 2(4), pages 553–589. doi :MR2119929. 2004.

[92] EFENDIEV Y.R. et WU X.H. Multiscale finite element for problems with highly oscillatory coefficients. *Numerische Mathematik*, 90, pages 459–486. doi :10.1007/s002110100274. 2002.

[93] Environnement Haute Performance pour l'Optimisation et la Conception. `http://www.systematic-paris-region.org/fr/UserFiles/File/EHPOC.pdf`.

[94] ERN A. et GUERMOND J.L. Theory and practice of finite elements, *Applied mathematical sciences*, tome 159. Springer. 2004.

[95] EYMARD R., GALLOUËT et RAPHAÈLE H. Finite volume methods. P.G. Ciarlet et J.L. Lions (rédacteurs), Solution of Equation in $\mathbb{R}^n$ (Part 3), Techniques of Scientific Computing (Part 3), *Handbook of Numerical Analysis*, tome 7, pages 713–1020. Elsevier. doi :10.1016/S1570-8659(00)07005-8. 2000.

[96] EYMARD R. et HÉRARD J.M. (rédacteurs). Finite Volumes for Complex Applications V, Problems & Perspectives. ISTE, Wiley. 2008.

[97] FAILLE I. A control volume method to solve an elliptic equation on a two-dimensional irregular mesh. *Comput. Methods Appl. Mech. Eng.*, 100(2), pages 275–290. doi :10.1016/0045-7825(92)90186-N. 1992.

[98] FEDERER H. Geometric measure theory. Springer-Verlag. 1969.

[99] FREY P.J. et GEORGE P.L. Mesh generation : application to finite elements. 2^{e} édition. Wiley-ISTE. 2008.

[100] GALIGNANI A. et GALIGNANI W. The History of Paris, from the earliest period to the present day. Galignani. 1825.

[101] GALLUCCI E., SCRIVENER K., GROSO A., STAMPANONI M. et MARGARITONDO G. 3D experimental investigation of the microstructure of cement pastes using synchrotron X-ray microtomography (μCT). *Cement and Concrete Research*, 37(3), pages 360–368. doi :10.1016/j.cemconres.2006.10.012. 2007.

[102] GALLÉ C. Effect of drying on cement-based materials pore structure as identified by mercury intrusion porosimetry : A comparative study between oven-, vacuum-, and freeze-drying. *Cement and Concrete Research*, 31(10), pages 1467–1477. doi :10.1016/S0008-8846(01)00594-4. 2001.

[103] Ghs3d : a powerful isotropic tet-mesher. `http://www-roc.inria.fr/gamma/gamma/ghs3d/ghs.php`.

[104] GILBARG D. et TRUDINGER N.S. Elliptic Partial Differential Equations of Second Order. Classics in Mathematics, 13^{e} édition. Springer. 1983.

[105] GINTING V. Analysis of two-scale finite volume element method for elliptic problem. *Journal of Numerical Mathematics*, 12(2), pages 119–141. 2004.

[106] GLORIA A. An analytical framework for the numerical homogenization of monotone elliptic operators and quasiconvex energies. *Multiscale Modeling & Simulation*, 5(3), pages 996–1043. doi :10.1137/060649112. 2006.

[107] GLORIA A. An analytical framework for numerical homogenization. Part II : Windowing and oversampling. *Multiscale Modeling & Simulation*, 7(1), pages 274–293. doi :10.1137/070683143. 2008.

[108] GLORIA A. Reduction of the resonance error. Part I : Approximation of homogenized coefficients. `http://hal.archives-ouvertes.fr/inria-00457159/fr/`. 2010.

[109] GNU Lesser General Public License. `http://www.gnu.org/licenses/lgpl-3.0.txt`.

[110] HAECKER C.J., GARBOCZI E., BULLARD J., BOHN R., SUN Z., SHAH S. et VOIGT T. Modeling the linear elastic properties of portland cement paste. *Cement and Concrete Research*, 35(10), pages 1948–1960. doi : 10.1016/j.cemconres.2005.05.001. 2005.

[111] HAJIBEYGI H., BONFIGLI G., HESSE M.A. et JENNY P. Iterative multiscale finite-volume method. *Journal of Computational Physics*, 227(19), pages 8604–8621. doi :10.1016/j.jcp.2008.06.013. 2008.

[112] HAJIBEYGI H. et JENNY P. Multiscale finite-volume method for parabolic problems arising from compressible multiphase flow in porous media. *Journal of Computational Physics*, 228(14), pages 5129–5147. doi : 10.1016/j.jcp.2009.04.017. 2009.

[113] HENSON V.E. et YANG U.M. BoomerAMG : A parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics*, 41(1), pages 155–177. doi :10.1016/S0168-9274(01)00115-5. 2002.

[114] HERBIN R. et HUBERT F. Benchmark on discretization schemes for anisotropic diffusion problems on general grids. `http://www.latp.univ-mrs.fr/latp_numerique/?q=node/3`. 2008.

[115] HEWLETT P. (rédacteur). Lea's Chemistry of Cement and Concrete. 4e édition. Butterworth Heinemann. 2003.

[116] HEXOTIC : an automated all-hexahedral mesher. `http://www-roc.inria.fr/gamma/gamma/Logiciels/Hexotic`.

[117] HIGGINS B. Experiments and observations made with the view of improving the art of composing and applying calcareous cements, and of preparing quick-lime. T. Cadell, London. 1780.

[118] HIRT C.W., AMSDEN A.A. et COOK J.L. An arbitrary Lagrangian-Eulerian computing method for all flow speeds. *Journal of Computational Physics*, 14(3), pages 227–253. doi :10.1016/0021-9991(74)90051-5. 1974.

[119] HOU T., WU X. et ZHANG Y. Removing the cell resonance error in the multiscale finite element method via a Petrov-Galerkin formulation. *Communications in Mathematical Sciences*, 2(2), pages 185–205. 2004.

[120] HOU T.Y. et WU X.H. A multiscale finite element method for elliptic problems in composite materials and porous media. *Journal of computational physics*, 134(1), pages 169–189. doi :10.1006/jcph.1997.5682. 1997.

[121] HOU T.Y., WU X.H. et CAI Z. Convergence of a multiscale finite element method for elliptic problems with rapidly oscillating coefficients. *Mathematics of Computation*, 68(227), pages 913–943. doi :10.1090/S0025-5718-99-01077-7. 1999.

[122] HUGHES T.J., SCOVAZZI G., BOCHEV P.B. et BUFFA A. A multiscale discontinuous galerkin method with the computational structure of a continuous galerkin method. *Computer Methods in Applied Mechanics and Engineering*, 195(19–22), pages 2761–2787. doi :10.1016/j.cma.2005.06.006. 2006.

[123] Hypre. `http://acts.nersc.gov/hypre/`.

[124] JANG G.W., KIM J.E. et KIM Y.Y. Multiscale galerkin method using interpolation wavelets for two-dimensional elliptic problems in general domains. *International Journal for Numerical Methods in Engineering*, 59(2), pages 225–253. doi :10.1002/nme.872. 2004.

[125] JENNY P., LEE S.H. et TCHELEPI H.A. Multiscale finite-volume method for elliptic problems in subsurface flow simulation. *Journal of Computational Physics*, 187(1), pages 47–67. doi :10.1016/S0021-9991(03)00075-5. 2003.

[126] JENNY P., LEE S.H. et TCHELEPI H.A. Adaptive multiscale finite-volume method for multiphase flow and transport in porous media. *Multiscale Modeling & Simulation*, 3(1), pages 50–64. doi :10.1137/030600795. 2005.

[127] JENNY P., LEE S.H. et TCHELEPI H.A. Adaptive fully implicit multiscale finite-volume method for multi-phase flow and transport in heterogeneous porous media. *Journal of Computational Physics*, 217(2), pages 627–641. doi :10.1016/j.jcp.2006.01.028. 2006.

[128] JENNY P. et LUNATI I. Modeling complex wells with the multi-scale finite-volume method. *Journal of Computational Physics*, 228(3), pages 687–702. doi :10.1016/j.jcp.2008.09.026. 2009.

[129] KARIM M. et KRABBENHOFT K. Extraction of effective cement paste diffusivities from x-ray microtomography scans. *Transport in Porous Media*, 84, pages 371–388. doi :10.1007/s11242-009-9506-y. 2010.

[130] KUMBARO A. et SEIGNOLE V. Two-phase flow computing with OVAP code. Workshop Trends in Numerical and Physical Modeling of Industrial Two-phase Flow. Cargèse, France. 2001.

[131] LAUG P. et BOROUCHAKI H. BLSURF - mailleur de surfaces composées de carreaux paramétrés - manuel d'utilisation. Rapport technique 0232, Inria. 1999.

[132] LE BRIS C. Systèmes multi-échelles : Modélisation et simulation, *Mathématique et Applications*, tome 47, chapitre 2. Springer. 2005.

[133] LE TALLEC P. Domain decomposition methods in computational mechanics. *Computational mechanics advances*, 1(2), pages 121–220. 1994.

[134] LEE S., WOLFSTEINER C. et TCHELEPI H. Multiscale finite-volume formulation for multiphase flow in porous media : black oil formulation of compressible, three-phase flow with gravity. *Computational Geosciences*, 12, pages 351–366. doi :10.1007/s10596-007-9069-3. 2008.

[135] LOVERA P., GALLÉ C. et LE BESCOP P. Towards an intrinsic relationship between diffusion coefficients and microscopic features of cements ? Scientific Basis for Nuclear Waste Managment XXIV, *Materials Research Society Symposium Proceedings*, tome 663, pages 81–88. 2000.

[136] LU S., LANDIS E. et KEANE D. X-ray microtomographic studies of pore structure and permeability in portland cement concrete. *Materials and Structures*, 39, pages 611–620. doi :10.1617/s11527-006-9099-7. 2006.

[137] LUNATI I. et JENNY P. Multiscale finite-volume method for compressible multiphase flow in porous media. *Journal of Computational Physics*, 216(2), pages 616–636. doi :10.1016/j.jcp.2006.01.001. 2006.

[138] LUNATI I. et JENNY P. Multiscale finite-volume method for density-driven flow in porous media. *Computational Geosciences*, 12, pages 337–350. 10.1007/s10596-007-9071-9. 2008.

[139] MADAY Y., MAVRIPLIS C. et PATERA A. Nonconforming mortar element methods : Application to spectral discretizations. Domain decomposition methods, Proceedings of the Second International Symposium, Los Angeles, CA ;United States, pages 392–418. 1989.

[140] MATACHE A.M., BABUŠKA I. et SCHWAB C. Generalized *p*-FEM in homogenization. *Numerische Mathematik*, 86(2), pages 319–375. doi : 10.1007/PL00005409. 2000.

[141] MATACHE A.M. et SCHWAB C. Two-scale fem for homogenization problems. *Mathematical Modelling and Numerical Analysis*, 36(4), pages 537–572. doi :10.1051/m2an:2002025. 2002.

[142] Med. `https://hammi.extra.cea.fr/static/MED/web_med/pdf/NOTE_HI-26-05-005-A.pdf`.

[143] MILTON G.W. The Theory of Composites. Cambidge Monographs on Applied and Computational Mathematics. doi :10.2277/0521781256. 2002.

[144] MING P. et ZHANG P. Analysis of the heterogeneous multiscale method for parabolic homogenization problems. *Mathematics of Computation*, 76(257), pages 153–177. doi :10.1090/S0025-5718-06-01909-0. 2007.

[145] MURAT F. et TARTAR L. Topics in the Mathematical Modelling of Composite Materials, *Progress in Nonlinear Differential Equations and Their Applications*, tome 31, chapitre H-Convergence, pages 21–44. Birkhäuser. 1997.

[146] MD Nastran : Integrated, multidiscipline CAE solution. `http://www.mscsoftware.com/Products/CAE-Tools/MD-Nastran.aspx`.

[147] Netgen - automatic mesh generator. `http://www.hpfem.jku.at/netgen`.

[148] NF P15-301 : Liants hydrauliques - Ciments courants - Composition, spécifications et critères de conformité. 1994.

[149] NX I-deas MasterFEM. http://www.plm.automation.siemens.com/fr_fr/products/nx/simulation/master_fem/index.shtml.

[150] ODLER I. Hydration, setting and hardening of portland cement. P. Hewlett (rédacteur), Lea's Chemistry of Cement and Concrete, 4e édition, chapitre 6, pages 241–297. Butterworth Heinemann. 2003.

[151] OHLBERGER M. A posteriori error estimates for the heterogeneous multiscale finite element method for elliptic homogenization problems. *Multiscale Modeling & Simulation*, 4(1), pages 88–114. doi :10.1137/040605229. 2005.

[152] OWHADI H. et ZHANG L. Homogenization of parabolic equations with a continuum of space and time scales. *SIAM Journal on Numerical Analysis*, 46, pages 1–36. doi :10.1137/060670420. 2007.

[153] OWHADI H. et ZHANG L. Metric-based upscaling. *Communications on Pure and Applied Mathematics*, 60(5), pages 675–723. 2007.

[154] Portable, Extensible Toolkit for Scientific Computation. http://www-unix.mcs.anl.gov/petsc/petsc-as/.

[155] Python programming language. http://www.python.org.

[156] RAVIART P. et THOMAS J. A mixed finite element method for 2-nd order elliptic problems. I. Galligani et E. Magenes (rédacteurs), Mathematical Aspects of Finite Element Methods, *Lecture Notes in Mathematics*, tome 606, pages 292–315. Springer Berlin / Heidelberg. doi :10.1007/BFb0064470. 1977.

[157] RAVIART P.A. et THOMAS J.M. Introduction à l'analyse numérique des équations aux dérivées partielles. Mathématiques appliquées pour la maîtrise. Dunod. 2004.

[158] REED W.H. et HILL T.R. Triangular mesh methods for the neutron transport equation. Rapport technique LA-UR-73-479, Los Alamos Scientific Laboratory. 1973.

[159] RICHET C., LE BESCOP P., GALLÉ C., PEYCELON H., BEJAOUI S., POINTEAU I., L'HOSTIS V. et BARY B. Dossier de synthèse sur le comportement à long terme des colis : dossier de référence phénoménologique "colis béton" 2004. Rapport technique DPC/SCCME/04-679-A, CEA, Département de Physico-Chimie. 2004.

[160] RIVIÈRE B., WHEELER M.F. et GIRAULT V. Improved energy estimates for interior penalty, constrained and discontinuous Galerkin methods for elliptic problems. part i. *Computational Geosciences*, 3(3), pages 337–360. doi : 10.1023/A:1011591328604. 1999.

[161] SAAD Y. Iterative methods for sparse linear systems. 2e édition. Society for Industrial Mathematics. 2003.

[162] SAAD Y. et SCHULTZ M.H. GMRES : A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3), pages 856–869. doi :10.1137/0907058. 1986.

[163] Plate-forme SALOME. `http://www.salome-platform.org/`.

[164] SANCHEZ-PALENCIA E. Non-Homogeneous Media and Vibration Theory, *Lecture Notes in Physics*, tome 127. Springer-Verlag. 1980.

[165] SCRIVENER K.L. Backscattered electron imaging of cementitious microstructures : understanding and quantification. *Cement and Concrete Composites*, 26(8), pages 935–945. doi :10.1016/j.cemconcomp.2004.02.029. Scanning electron microscopy of cements and concretes. 2004.

[166] SCRIVENER K.L., CRUMBIE A.K. et PRATT P.L. A study of the interfacial region between cement paste and aggregate in concrete. S. Mindess et S.P. Shah (rédacteurs), Bonding in Cementitious Composites, *MRS Proceedings*, tome 114. 1988.

[167] SÉCHER B. Numerical platon users guide and reference manual v3.0. Rapport technique SFME/LGLS/RT/01-001, CEA, DEN/DM2S. 2006.

[168] STAUFFER D. et BUNDE A. Introduction to Percolation Theory, tome 40. AIP. doi :10.1063/1.2820231. 1987.

[169] STRANG G. et FIX G.J. An analysis of the finite element method. Prentice-Hall Series in Automatic Computation. Prentice-Hall, Inc. 1973.

[170] TARTAR L. The General Theory of Homogenization, A Personalized Introduction, *Lecture Notes of the Unione Matematica Italiana*, tome 7. Springer. 2010.

[171] Trio-U. `http://www-trio-u.cea.fr/`.

[172] VICAT L.J. Recherches expérimentales sur les chaux de construction, les bétons et les mortiers ordinaires. Goujon. 1818.

[173] WOHLMUTH B.I. A mortar finite element method using dual spaces for the Lagrange multiplier. *SIAM Journal on Numerical Analysis*, 38, pages 989–1012. doi :10.1137/S0036142999350929. 2000.

[174] WOLFSTEINER C., LEE S.H. et TCHELIPI H.A. Well modeling in the multiscale finite volume method for subsurface flow simulation. *Multiscale Modeling & Simulation*, 5(3), pages 900–917. doi :10.1137/050640771. 2006.

[175] XINGYE Y. et WEINAN E. The local microscale problem in the multiscale modeling of strongly heterogeneous media : Effects of boundary conditions and cell size. *Journal of Computational Physics*, 222(2), pages 556–572. doi :10.1016/j.jcp.2006.07.034. 2007.

[176] XUEQAN W., DONGXU L., XIUN W. et MINSHU T. Modification of the interfacial zone between aggregate and cement paste. S. Mindess et S.P. Shah (rédacteurs), Bonding in Cementitious Composites, *MRS Proceedings*, tome 114. 1988.

[177] ZHOU H. et TCHELEPI H.A. Operator-based multiscale method for compressible flow. *SPE Journal*, 13(3), pages 267–273. doi :10.2118/106254-PA. 2008.

Résumé.

Les méthodes numériques classiques sont inadaptées aux problèmes de diffusion au sein des matériaux cimentaires, du fait de l'écart entre l'échelle grossière de travail, de l'ordre du mètre, et l'échelle fine de la description du milieu, de l'ordre du micromètre, On présente dans cette thèse plusieurs méthodes de simulations multi-échelles couplant Volumes Finis et Éléments Finis, ainsi que leurs implémentations informatiques efficaces.

En particulier, on a développé une chaîne de calcul multi-échelle utilisant la plate-forme *SALOME* (génération des maillages, post-traitement des solutions) et le code de calcul parallèle *MPCube* (résolution des problèmes) qui permet de réaliser automatiquement et efficacement des simulations multi-échelles. Un soin particulier a été apporté à la parallélisation des tâches et à l'optimisation des méthodes aux spécificités des matériaux cimentaires.

On a appliqué cette chaîne de calcul à plusieurs échantillons de matériaux cimentaires, notamment des modèles de mortiers et de pâtes de ciment. Les résultats de ces simulations ont permis de déterminer une diffusivité numérique équivalente, ainsi que de reconstruire une solution à l'échelle fine.

Abstract.

To solve diffusion problems on cement media, two scales must be taken into account : a fine scale, which describes the micrometers wide microstructures present in the media, and a work scale, which is usually a few meters long. Direct numerical simulations are almost impossible because of the huge computational resources (memory, CPU time) required to assess both scales at the same time. To overcome this problem, we present in this thesis multiscale resolution methods using both Finite Volumes and Finite Elements, along with their efficients implementations.

More precisely, we developped a multiscale simulation tool which uses the *SALOME* platform to mesh domains and post-process data, and the parallel calcul code *MPCube* to solve problems. This *SALOME-MPCube* tool can solve automatically and efficiently multi-scale simulations. Parallel structure of computer clusters can be use to dispatch the most time-consuming tasks. We optimized the functions to take into account for cement media specificities.

We present numerical experiments on various cement media samples, e.g. mortar and cement paste. From these results, we manage to compute a numerical effective diffusivity of our cement media and to reconstruct a fine scale solution.