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 :

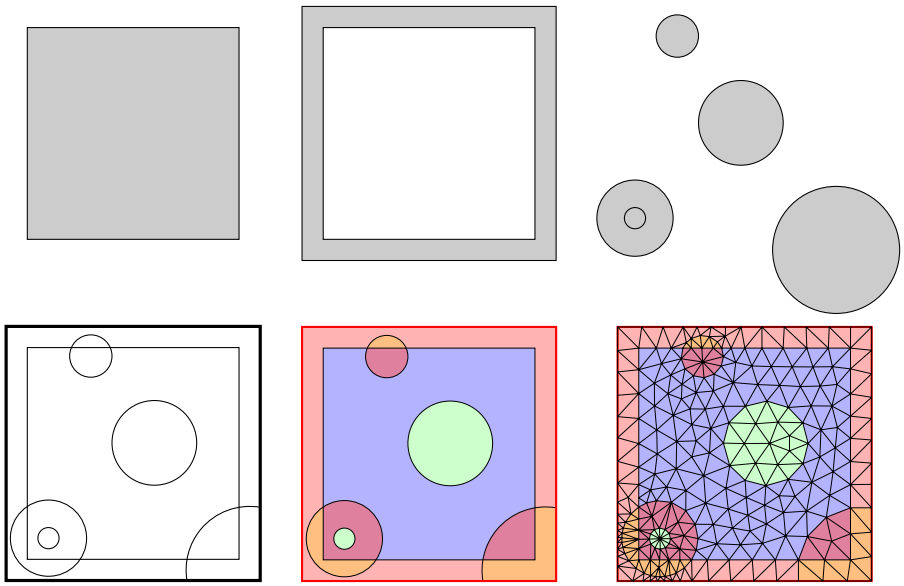

FIGURE 6.1 – Illustration du travail de l'Algorithme 6.1 sur une cellule. En haut, de gauche à droite, les objets indépendants : le macroélément homogène, la couronne homogène et les inclusions. En bas, de gauche à droite, la cellule assemblée en un unique objet, le marquage des groupes d'objets (chaque 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.

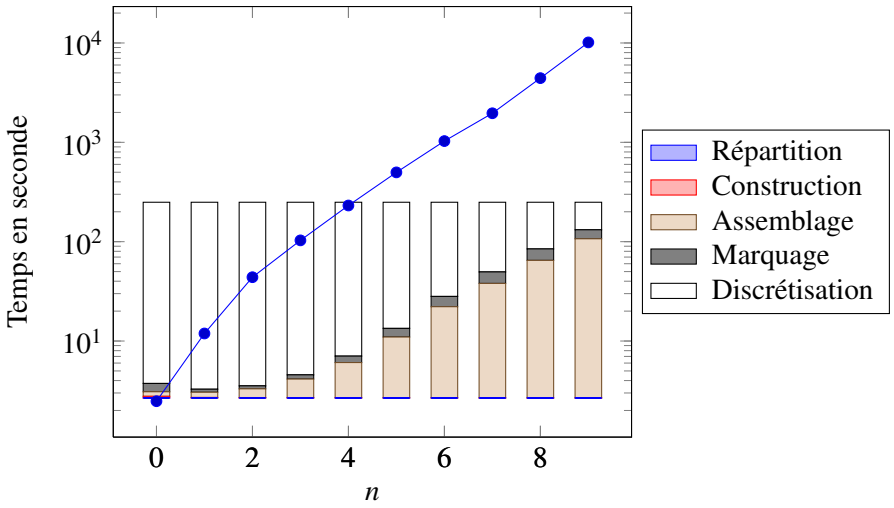

FIGURE 6.2 – Temps de génération, en secondes, des maillages $T_h(\Omega_n)$ en fonction de n. Les barres présentent la répartition du temps total sur les différentes étapes de construction. L'étape d'assemblage apparaît clairement comme étant la plus chronophage.

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.

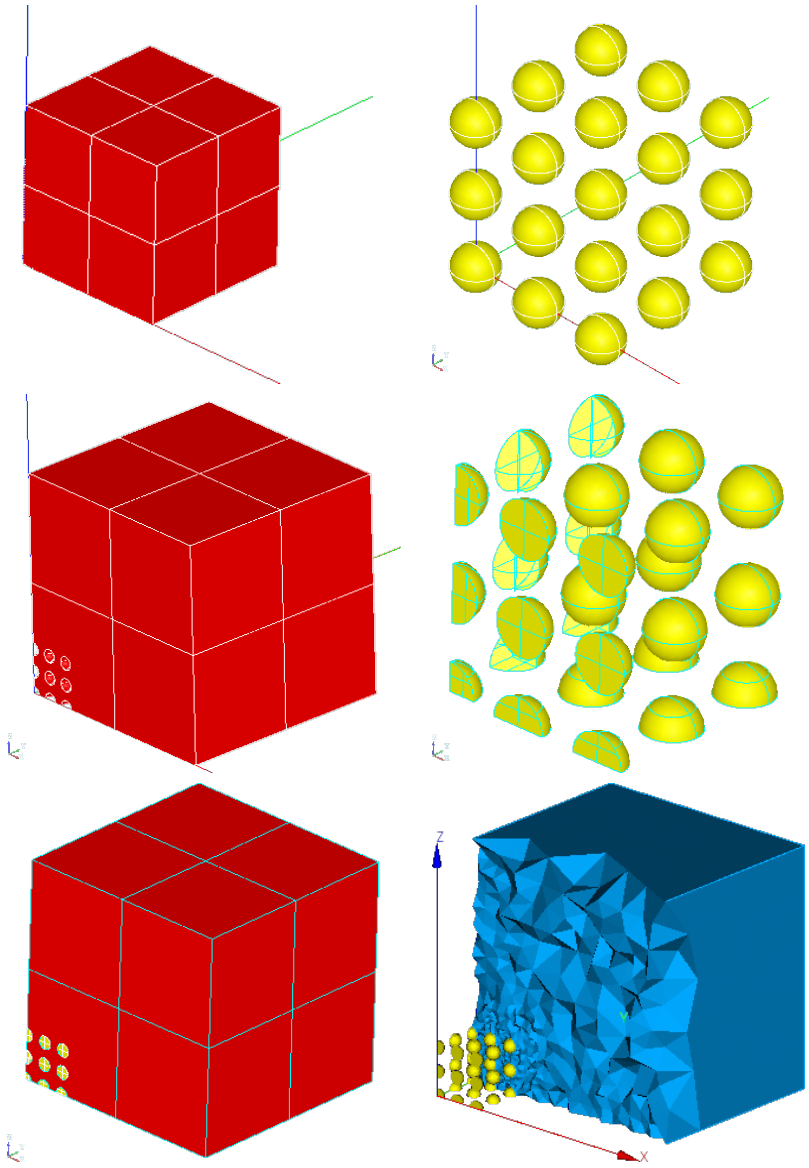

FIGURE 6.3 – Captures d'écran de la plate-forme *SALOME* lors du travail de l'Algorithme 6.1 sur l'Exemple 6.1 pour $n = 2$. L'assemblage de la géométrie de la cellule a été détaillée afin d'illustrer le travail de l'Algorithme 6.3. De gauche à droite et de bas en haut : la zone homogène de macroélément 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

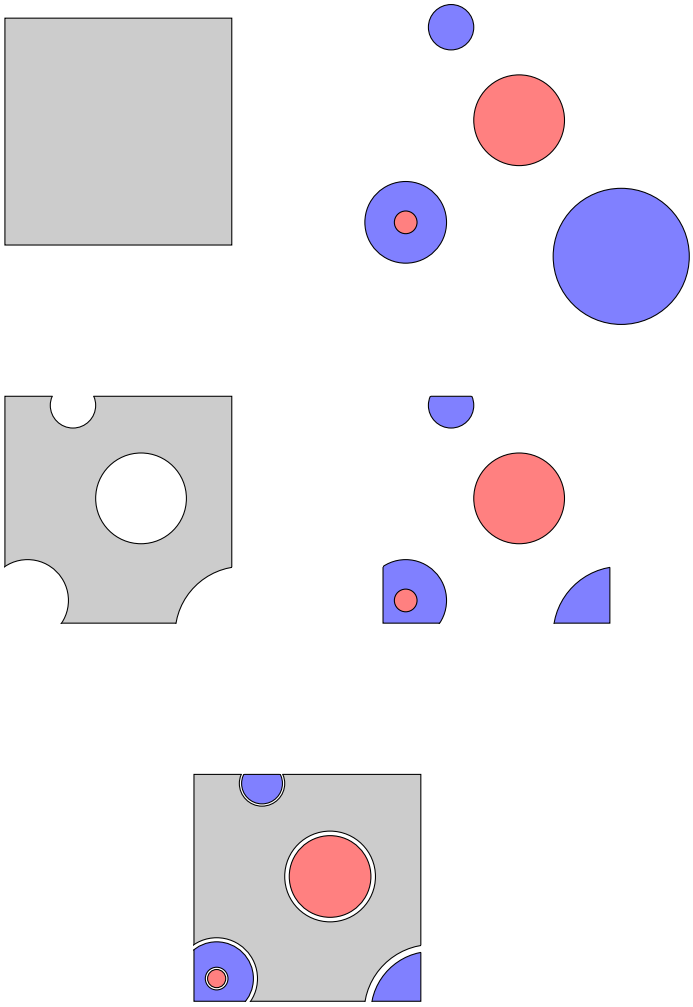

FIGURE 6.4 – Illustration de l'assemblage de la géométrie d'une cellule par l'Algorithme 6.3. Seul le travail sur le macroélément est ici représenté, la zone de recouvrement étant traitée de manière similaire. De gauche à droite et de haut en bas : la zone homogène de macroélément K_0 (en gris), les 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 :

– $\mathscr{I}^K_M$: ensemble des inclusions *strictement* intérieures au macroélément.

$$\mathscr{I}^K_M = \left\{ I \in \mathscr{I}^K, \mathbb{B}(I) \subset K_0 \right\}$$

– $\mathscr{I}^K_{MC}$: ensemble des inclusions à cheval sur le macroélément et la couronne.

$$\mathscr{I}^K_{MC} = \left\{ I \in \mathscr{I}^K, \mathbb{B}(I) \cap K_0 \neq \emptyset, \mathbb{B}(I) \cap K_0^c \neq \emptyset \right\}$$

– $\mathscr{I}^K_C$: ensemble des inclusions *strictement* intérieures à la couronne.

$$\mathscr{I}^K_C = \left\{ I \in \mathscr{I}^K, \mathbb{B}(I) \subset K_0^c \right\}$$

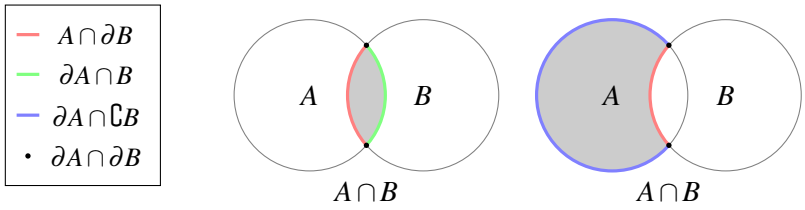

FIGURE 6.5 – Intersection (à gauche) et coupe (à droite) de deux ensembles A et B. On distingue les différentes surfaces caractéristiques utilisées par l'Algorithme 6.4 et l'Algorithme 6.5.

– $\mathscr{I}^K_{CE}$: 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 :

$$\begin{aligned}\partial(A\cap B)\cap\partial A = {} & (\partial(A\cap B)\cap\partial A\cap B)\\ & \sqcup(\partial(A\cap B)\cap\partial A\cap\partial B)\sqcup\left(\partial(A\cap B)\cap\partial A\cap\complement B\right).\end{aligned}$$

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.*

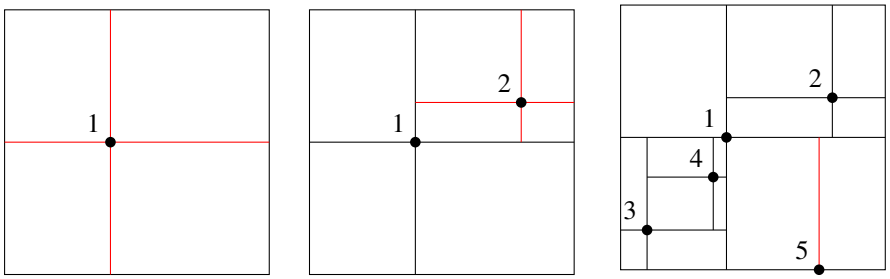

FIGURE 6.6 – Ajout de points à une face par l'Algorithme 6.6. On ajoute d'abord le premier point de tangence (à gauche), puis le second (au milieu), jusqu'à obtenir l'ensemble des points (à droite).

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.

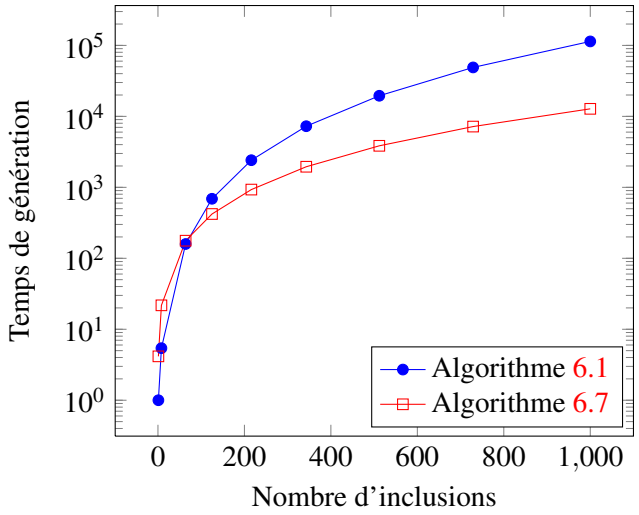

FIGURE 6.7 – Temps nécessaires à la génération de la géométrie du milieu Ω_n, en fonction du nombre d'inclusions $N = (n+1)^3$. Les temps présentés ont été moyennés sur 10 construction et on a choisi le temps moyen nécessaire à l'Algorithme 6.1 pour générer Ω_0 comme unité de référence.

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

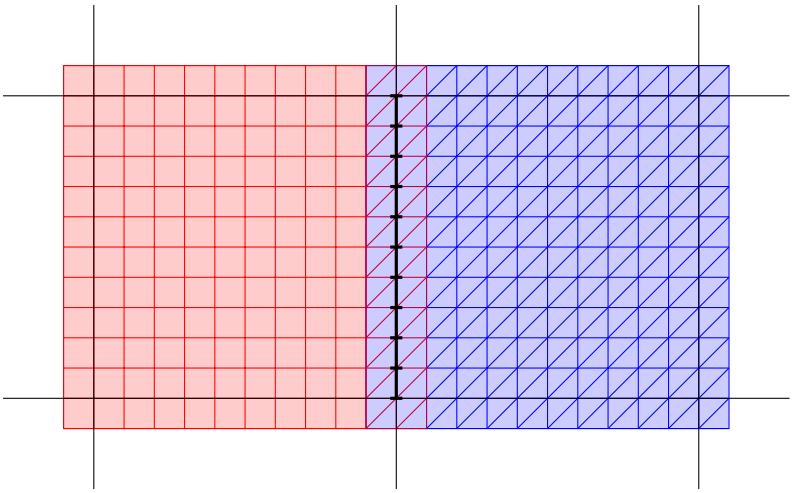

FIGURE 6.8 – Maillages coïncidents : on construit les cellules $\hat{K}^-$ (en rouge) et $\hat{K}^+$ (en bleu) de deux macroéléments voisins K^- et K^+. Les discrétisations $T_h(\hat{K}^-)$ et $T_h(\hat{K}^+)$ sont distinctes, et certaines zones du milieu ont donc deux discrétisations. Cependant la discrétisation 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.

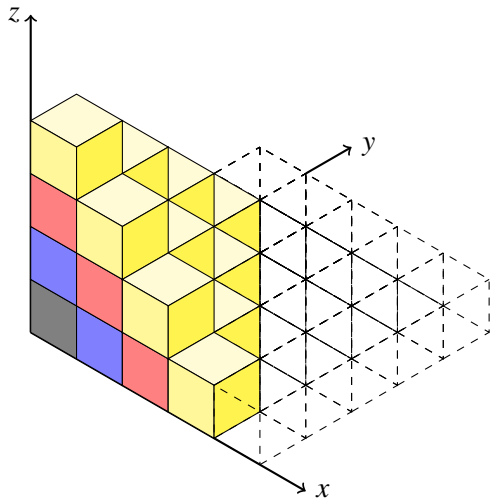

FIGURE 6.9 – Maillages coïncidents par la méthode des jeux successifs. Pour obtenir des maillages fins coïncidents, les cellules sont traitées de proche en proche à partir de la cellule $(0,0,0)$ (en gris). Le maillage de chaque cellule est construit en tenant compte des surfaces maillées dans les jeux 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.

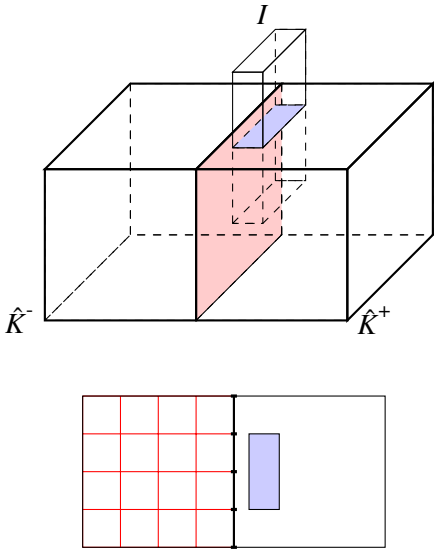

FIGURE 6.10 – Maillages coïncidents par la méthode des jeux successifs.
En haut : deux cellules voisines $\hat{K}^-$ (à gauche) et $\hat{K}^+$ (à droite), pour lesquelles on a pris $\rho = 0$, et une inclusion I de $\hat{K}^+$ située près de $\partial K = \partial K^- \cap \partial K^+$ (en rouge).
En 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

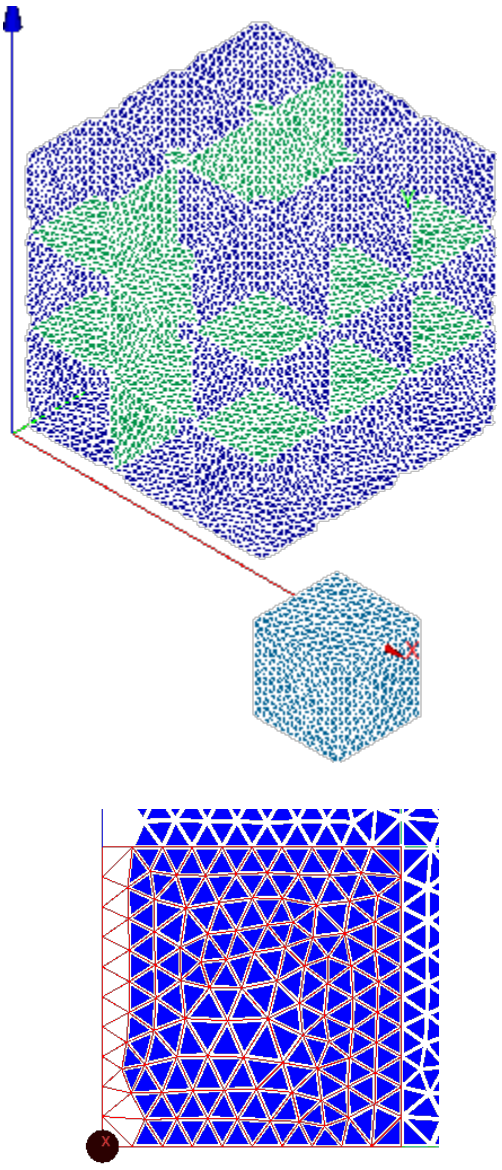

FIGURE 6.11 – Maillages coïncidents par maillage de peau : captures d'écran de la plate-forme *SALOME*.
En haut : le maillage de référence pour le découpage $(\mathscr{D},\rho) = (\{3,3,3\},0)$ et le maillage de la cellule $\hat{K} = (2,0,0)$. La frontière $\partial\Omega$ du milieu Ω n'est pas représentée 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.

■

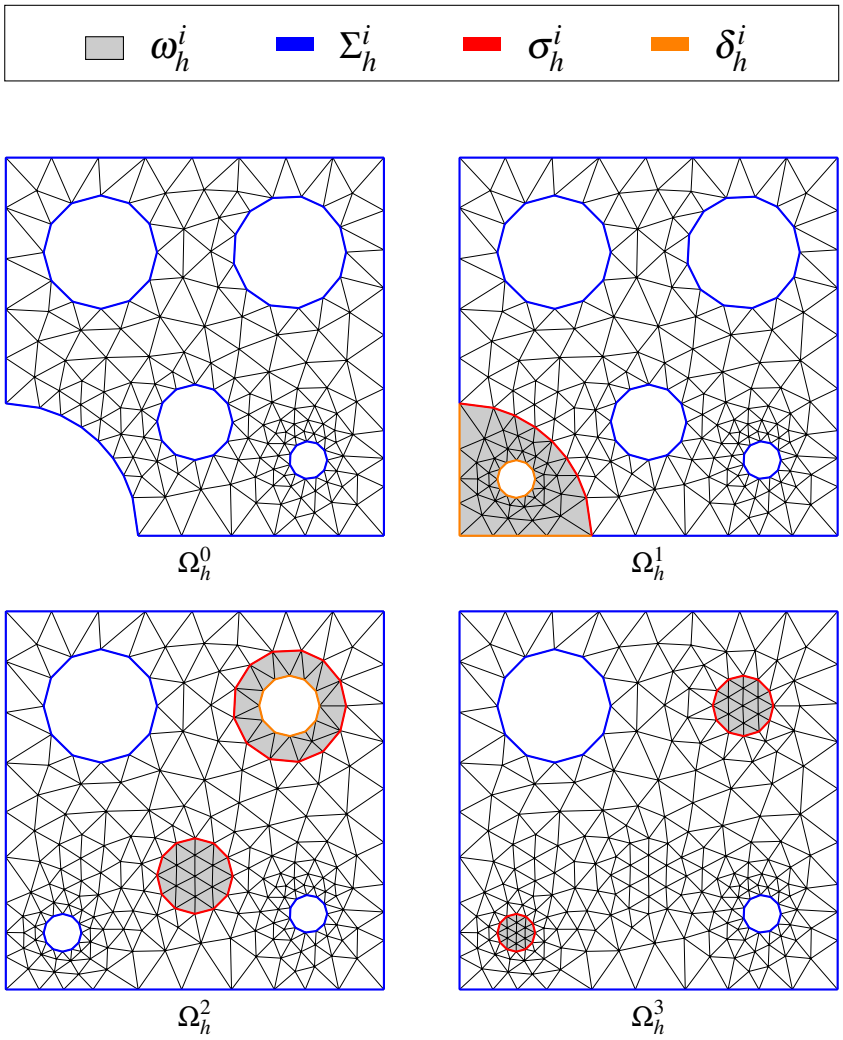

FIGURE 6.12 – Adaptation de la discrétisation d'un milieu au cours de sa dégradation. À partir de la discrétisation Ω_h^3, correspondant au milieu le plus dégradé, l'Algorithme 6.10 construit les discrétisations Ω_h^2, Ω_h^1 et Ω_h^0 en suivant le processus de dégradation à rebours. On a fait ici apparaître en couleurs les différents ensembles utilisés par l'Algorithme 6.10.