UNIVERSITÉ DE LA MEDITERRANEE
ÉCOLE DOCTORALE DE MATHÉMATIQUES ET INFORMATIQUE E.D. 184

THÈSE

présentée pour obtenir le grade de

Docteur de l'Université de la Méditerranée.

Spécialité : Informatique et Mathématiques

par

Thomas BONACCORSI

sous la direction du Pr. Marc DANIEL

Titre :

Modélisation pluridisciplinaire d'expériences d'irradiation dans un réacteur d'irradiation technologique.

soutenue publiquement le 21 septembre 2007

JURY

Pr. Dominique FAUDOT	Université de Bourgogne	*Rapporteur*
Pr. Abdelaziz BOURAS	Université de Lyon 2	*Rapporteur*
Pr. Marc DANIEL	Université de la Méditerranée	*Directeur*
Pr. Jean Koclas	Université de Montréal	*Examinateur*
Dr. Thierry N'KAOUA	Commissariat à l'énergie atomique	*Examinateur*
Dr. Marc GRANDOTTO	Commissariat à l'énergie atomique	*Examinateur*
Dr. Jacques DI SALVO	Commissariat à l'énergie atomique	*Examinateur*

A Elodie,
A mes parents et à Agathe
A Michelle et à la mémoire de Jean-Louis Plantier.

Remerciements

Les professeurs Dominique Faudot de l'Université de Bourgogne et Abdelaziz Bouras de l'Université de Lyon 2 ont toute ma reconnaissance pour avoir accepté d'être les rapporteurs de ma thèse. Etant tous deux informaticiens, je suis conscient du difficile travail de rapporteur et les remercie pour leurs efforts de lecture de nombreux paragraphes de physique.

Merci au docteur Marc Grandotto, du Service de Simulation en Thermohydraulique (SSTH) pour son suivi et ses conseils précieux, ainsi que pour m'avoir impliqué dans les projets de couplage en collaboration EDF/CEA. Je le remercie également d'être examinateur de ce jury de thèse.

Je remercie le professeur Jean Koclas, directeur de l'Institut de Génie Nucléaire de l'école Polytechnique de Montréal, pour avoir accepté de faire partie de ce jury de thèse en dépit de la distance qui sépare nos continents.

Je remercie également le docteur Thierry N'Kaoua, directeur de la Gestion, des Contrats et des Ressources Humaines de la Direction de l'Energie Nucléaire du CEA, pour l'attention qu'il porte à mon travail en acceptant d'être examinateur.

Je désire exprimer toute ma gratitude au professeur Marc Daniel de l'Université de la Méditerranée, pour son rôle de directeur de thèse, son investissement sans faille et la disponibilité qu'il m'a accordée quel que soit le moment. Son suivi quasi quotidien et ses encouragements m'ont rassuré et m'ont guidé tout au long de ces trois années de thèse.

Toute ma reconnaissance va envers Jacques Di-Salvo mon encadrant CEA. Il a pu me guider et me conseiller tout en me transmettant son métier de neutronicien. Il m'a fait confiance pour nombre de mes choix et m'a toujours apporté son soutien, même au-delà de son rôle d'encadrant. Sa bonne humeur a rendu plus agréable ces trois années exceptionnelles. Il est pour beaucoup dans l'accomplissement de ce travail.

Même si nous étions éloignés géographiquement, Guy Willermoz, qui est l'initiateur du sujet de thèse, m'a suivi de près et s'est beaucoup impliqué dans les travaux réalisés. C'est sans compter qu'il s'est investi et qu'il a fait connaître mon travail partout où il est passé. De même Patricia Sireta s'est énormément investie et m'a intégré dans l'équipe dès le commencement de la thèse. En m'informant des projets en cours, elle m'a permis de participer pleinement à la vie du laboratoire. Tous deux ont complété l'encadrement de Jacques en s'assurant que les choses allaient pour le mieux et ont été les relais vers les nombreux contacts que j'ai eus pour mener au mieux ce projet. Je les en remercie chaleureusement.

Cette thèse a été réalisée au Laboratoire de Projet Nucléaire (LPN) du Service de Physique des Réacteurs et du Cycle (SPRC). Je tiens pour cela à remercier Alain Zaetta qui m'a accueilli en tant que chef de service, ainsi que Jean-Paul Grouiller chef du SPRC. Je

remercie également Bruno Maugard qui a été le chef du LPN durant les deux premières années de ma thèse et Jean-Pascal Hudelot actuellement chef du LPN. Les moyens techniques et logistiques dont j'ai bénéficié durant la thèse m'ont été d'une grande aide.

Je remercie l'ensemble des personnes qui ont participé à ce travail de près ou de loin durant ces trois années :

- Jean Michel Ruggieri et Cyrille de Saint Jean n'ont pas hésité à me consacrer de leur temps pour me présenter, en début de thèse, leurs travaux réalisés sur des problèmes analogues aux miens. Ils m'ont mis en contact avec les personnes du groupe PAL. Ce fut le point de départ des réflexions de ce travail.

- Christophe Suteau m'a donné de son temps en me présentant le code ERANOS et m'a permis d'apporter ma contribution au Workshop ERANOS 2006.

- David Plancq et Gilles Thouvenin se sont penchés sur mon sujet avec beaucoup d'intérêt et grâce à leurs efforts j'ai réalisé un premier couplage avec les outils développés pendant la thèse.

- Jean-Marc Ricaud m'a fourni une première version de SALOME

- Nadir Bouhamou m'a donné des conseils sur SALOME et le format MED. Il a passé du temps avec moi pour installer le code PLEIADES et l'application ALCYONE sur ma machine.

- Patrick Obry, chef du Laboratoire de Simulation du Comportement des combustibles (LSC) du Service d'Etudes et de Simulation du Comportement des combustibles (SESC), a accepté de mettre à ma disposition le code PLEIADES et l'application ALCYONE.

- Martine Paolillo et Christian Caremoli de EDF SYNETICS, membres du groupe de travail PAL, m'ont conseillé et fourni des versions projets de leurs développements.

- Anne Nicolas, chef du LTSD du Service d'Etude des Réacteurs et de Mathématiques Appliquées (SERMA), a accepté de mettre à ma disposition les sources du logiciel Silene.

- Zarko Stankovski, auteur de Silene, a pris le temps de répondre à mes questions. Sans cela, l'exercice d'application de la thèse aurait été plus délicat à mettre en œuvre.

- Laurent Dada, chef du LGLS du Services Fluides numériques, Modélisation et Etudes (SFME), s'est intéressé à mes besoins techniques.

- Marc Tajchman m'a présenté la plate-forme SALOME, et Erwan Adam a été disponibile et a fait preuve d'efficacité pour résoudre mes problèmes avec XDATA.

- Frédéric Ducros, chef du Laboratoire de Modélisation et de Développement de Logiciels (LMDL) du Service de Simulation en ThermoHydraulique (SSTH) m'a permis de suivre une formation au code TRIO_U et s'est intéressé à mon travail. Gauthier Fauchet m'a aidé efficacement pour réaliser des calculs TRIO_U à partir de SALOME et

m'a donné des astuces de développement avec XDATA. Malheureusement, le travail de thermohydraulique qu'ils m'ont permis de réaliser n'est que peu représenté dans ce manuscrit. Il m'a cependant permis de cerner le problème de couplage dans son ensemble.

- Eric Royer m'a impliqué dans le groupe de travail PAL/Discipline.

- Philippe Dufour chef du Laboratoire d'Etude des Systèmes Avancés (LESA) du Service d'Etudes des Systèmes Innovants (SESI), Damien Moulin, Nicolas Schmidt et Laurence Buffe m'ont présenté leurs travaux et l'utilisation des codes FLICA et CASTEM.

- Helene Faure-Geors et Marie-Christine Anselmet du SESC m'ont apporté des informations sur le code METEOR et le comportement des crayons combustibles de TANOXOS.

- Michel Auclair, Xavier Wohleber et Fadhel Malouch du Service d'Irradiations en Réacteurs et d'Etudes Nucléaires (DRSN/SIREN) m'ont apporté les informations sur le dispositif et les conditions d'irradiations de TANOXOS, nécessaires à l'exercice d'application de la thèse.

Je tiens à remercier Corinne D'Aletto, Bernard Pouchin, Christophe Döderlein et Olivier Guéton pour l'attention qu'ils ont apporté à mon travail avec toutes les personnes qui m'ont entouré pendant ces trois années, en particulier Mireille Michez, Richard Girieud, Danielle Caer, Jean-Marc Palau, Gilles Youinou, Nicolas Huot, Thomas Cuvillier, Raphaël Enderle, Christian Jammes, Sophie Brochard et une attention particulière à François Barran qui s'est démené pour trouver des réponses à mes requêtes informatiques et pour me tenir compagnie lors des longues fins de journées. Je remercie également Sylvie Briaux secrétaire du SPRC et Sylvie Malleval secrétaire du LPN pour les aides précieuses qu'elles ont pu m'apporter et pour leur sympathie.

Merci à Gilles Gesquières de l'IUT d'Arles qui s'est intéressé à mes travaux et à Raphaël La Greca qui m'a offert la possibilité de m'essayer dans un rôle d'enseignant.

Je remercie aussi les copains Jeremy Koubbi, Van Tran Nam, Guillaume Thibault, Cédric Clouchoux, Adrien Nisan et Gregory Operto qui ont été ou seront bientôt en phase de rédaction, ainsi que mes amis qui se reconnaîtront.

Résumé

Un réacteur d'irradiation technologique permet d'irradier des échantillons de matériaux sous des flux neutroniques et photoniques intenses. Ces expériences sont réalisées dans des dispositifs expérimentaux localisés soit au coeur du réacteur, soit en périphérie (réflecteur).

Les outils de simulation dont disposent les physiciens ne traitent la plupart du temps qu'une seule discipline de manière très précise. C'est pourquoi les simulations multiphysiques des expériences d'irradiation nécessitent l'utilisation séquentielle de plusieurs codes de calcul et des échanges de données entre ces codes : il s'agit de problèmes de couplage.

Afin de développer des simulations multiphysiques pour les réacteurs OSIRIS et RJH, les travaux de thèse ont mis en place un modèle de données à partir d'objets informatiques que nous avons appelés Entités Technologiques. Ce modèle, commun à chacune des disciplines, permet de définir la géométrie d'un dispositif d'irradiation de manière paramétrique et d'y associer des informations sur les matériaux le composant.

Les simulations numériques sont encapsulées dans des classes informatiques présentant des interfaces (des méthodes et des attributs) spécifiques. De cette manière, les fonctionnalités nécessaires à l'échange de données et aux contrôles de l'exécution des calculs sont accessibles à partir des mêmes commandes informatiques (Mettre en données, lancer un calcul, post-traiter, récupérer des résultats...), quelle que soit la simulation numérique traitée. Ainsi, une fois encapsulées, ces simulations numériques peuvent être réutilisées et échangées pour différentes études.
Ce modèle de données est développé sous la forme d'un composant de la plate-forme SALOME. L'ensemble des études peut alors être réalisé dans SALOME.

Le premier cas d'application a permis de traiter le cas des simulations neutroniques (réacteurs OSIRIS et RJH) couplées avec des simulations du comportement du combustible. A terme, la thermohydraulique pourra également être prise en compte.
Outre l'amélioration de la précision de calcul dû au couplage des phénomènes de physique, la durée du développement de telles simulations se trouve fortement réduite et la plate-forme de simulation ouvre la possibilité du traitement des incertitudes.

mots-clés : dispositif expérimental, SALOME, couplage, calcul neutronique, modèle géométrique, échange de données.

Summary

A Material Testing Reactor (MTR) makes it possible to irradiate material samples under intense neutron and photonic fluxes. These experiments are carried out in experimental devices localised in the reactor core or in periphery (reflector).

Available physics simulation tools only treat, most of the time, one physics field in a very precise way. Multiphysic simulations of irradiation experiments therefore require a sequential use of several calculation codes and data exchanges between these codes: this corresponds to problems coupling.

In order to facilitate multiphysic simulations, this thesis sets up a data model based on data-processing objects, called Technological Entities. This data model is common to all of the physics fields. It permits defining the geometry of an irradiation device in a parametric way and to associate informations about materials to it.
Numerical simulations are encapsulated into interfaces providing the ability to call specific functionalities with the same command (to initialize data, to launch calculations, to post-treat, to get results, ...). Thus, once encapsulated, numerical simulations can be re-used for various studies.
This data model is developed in a SALOME platform component.

The first application case made it possible to perform neutronic simulations (OSIRIS reactor and RJH) coupled with fuel behavior simulations. In a next step, thermal-hydraulics could also be taken into account.
In addition to the improvement of the calculation accuracy due to the physical phenomena coupling, the time spent in the development phase of the simulation is largely reduced and the possibilities of uncertainty treatment are under consideration.

keywords: experimental device, SALOME, coupling, neutronic calculation, geometric model, data exchange.

Table des matières

Résumé **9**

Summary **11**

Introduction **25**

I Contexte et enjeux de la thèse **29**

1 Irradiations dans les réacteurs expérimentaux **31**

1.1 Comportement des matériaux sous irradiation 31

1.1.1 Matériaux de structure 31

1.1.2 Combustible nucléaire 32

1.2 Réacteur OSIRIS 35

1.2.1 Description du cœur 36

1.2.2 Dispositifs en cœur 37

1.2.3 Dispositifs en réflecteur 38

1.3 Réacteur Jules Horowitz 38

1.3.1 Description du cœur 38

1.3.2 Dispositifs en cœur 40

2 La simulation multidisciplinaire **43**

2.1 Maillages et codes de calcul 43

2.1.1 Méthodes de résolution de problèmes de physique 44

2.1.2 Maillages des méthodes de calcul déterministes 47

2.2 Principes de neutronique 51

2.2.1 Equation de BOLTZMANN 51

2.2.2 Principes de résolution 52

2.2.3 Influence des autres disciplines 53

2.3 Principes de thermohydraulique 55

2.3.1 Equation de Navier-Stokes 55
2.3.2 Principes de résolution 56
2.3.3 Influence des autres disciplines 57
2.4 Notions générales sur le couplage 58

3 Modèles géométriques 61
3.1 Modèle B-Rep 61
3.1.1 Principes élémentaires 61
3.1.2 Validité d'un modèle B-Rep 63
3.1.3 Synthèse 64
3.2 Modèle CSG 64
3.2.1 Principes élémentaires 64
3.2.2 Propriétés 65
3.2.3 Synthèse 67
3.3 Conversions de modèles 67
3.3.1 D'un modèle CSG vers un modèle B-Rep 67
3.3.2 D'un modèle B-Rep vers un modèle CSG 68
3.4 Modeleur sous contraintes 68
3.4.1 Modélisation paramétrique 69
3.4.2 Modélisation variationnelle 70
3.5 Modélisation par entités 71
3.5.1 Définition d'une entité 71
3.5.2 Utilisation 72
3.5.3 Synthèse 72

4 Environnement SALOME 73
4.1 La plate-forme SALOME 74
4.2 Le format MED 75
4.3 Composant SALOME 76
4.4 XDATA 77
4.5 Les Objets Technologiques 78
4.5.1 Intérêts et principes 78
4.5.2 Exemples de modélisation par Objets Technologiques 80
4.6 Discussion sur SALOME et son contexte logiciel 82

Synthèse de la première partie 85

II Mise en œuvre de la plate-forme de modélisation des dispositifs 87

5 Evaluation des Objets Technologiques 91

5.1 Modélisation du CHOUCA avec les Objets Technologiques 91

5.2 Discussion sur les Objets Technologiques 92

6 Définition du modèle de données 95

6.1 Objectifs et contraintes du modèle de données 95

6.1.1 Besoins pour les maillages et codes de calcul 95

6.1.2 Besoins en géométrie 97

6.1.3 Discussions sur les développements de la thèse 99

6.2 Proposition d'une méthode d'échange de grandeurs 103

6.2.1 Deux phases d'échanges de données 103

6.2.2 Schéma d'échange par maillage de référence 104

6.2.3 Schéma d'échange par interface 105

6.2.4 Discussion sur les schémas d'échanges 107

7 Mise en œuvre de l'architecture 109

7.1 Les Entités Technologiques 109

7.1.1 Généralités 110

7.1.2 Principes de modélisation par Entités Technologiques 111

7.1.3 Les formes géométriques 113

7.1.4 Le positionnement et les transformations géométriques 115

7.1.5 Les séquences de calculs 116

7.1.6 La gestion de versions des arbres d'Entités Technologiques 122

7.2 Le composant TECHENTITY 122

7.2.1 Les fonctionnalités Géométriques 122

7.2.2 La construction d'un arbre d'Entités Technologiques 123

7.2.3 La mise en œuvre d'une séquence de calculs 125

Synthèse de la deuxième partie 127

III Application à l'étude de dispositifs expérimentaux 129

8 Objectifs 133

8.1 Présentation du dispositif TANOXOS 133

8.2 Modélisation actuelle 134

8.3 Présentation du couplage envisagé 135
8.3.1 Modèle d'Entités Technologiques 136
8.3.2 Modélisation Neutronique du dispositif 137
8.3.3 Modélisation du comportement des crayons du dispositif 138
8.3.4 Couplage envisagé 139

9 Etude Neutronique 143
9.1 Modèle physique 143
9.2 Intégration 145
9.2.1 Pré-traitement 145
9.2.2 Post-traitement 149

10 Séquencement des disciplines 153
10.1 Encapsulation de la neutronique 153
10.2 Encapsulation du comportement combustible 154
10.3 Séquencement 154

Conclusion et perspectives 157

A Etude d'un crayon REP sous irradiation 167
A.1 Transferts thermiques dans le combustible 167
A.2 Transferts thermiques entre le crayon et le fluide 168
A.3 Température introduite dans les calculs neutroniques 169
A.4 Effets de contre-réaction 170

B Notions de section efficace 171

C Norme STEP et protocole d'application 214 173
C.1 Généralité 173
C.2 Les protocoles d'application 174
C.3 L'interface standard d'accès aux données (SDAI) 174
C.4 Le protocole d'application 214 176

D Utilisation de XDATA 181
D.1 Le concept XDATA 181
D.2 Les XType 182
D.3 Les XObject 183
D.4 Contruction d'un composant SALOME avec XDATA 184

E Description des principales matrices de transformations géométriques 187

F Description détaillée d'une classe Circle. 189

G Création d'un cube à l'aide du composant GEOM 193

H Description des menus du composant TECHENTITY. 197

I Création d'un cube à l'aide des objets Shape du composant TECHENTITY 203

J Utilisation d'un objet PyGeomFile 205

K Modèle d'Entités Technologiques de TANOXOS 207

L Validation numérique avec TRIPOLI4 du calcul neutronique présenté au chapitre 9 211

M Importation de fichier STEP dans SILENE 215

M.1 Fonctionnalités ajoutées 215

M.2 Utilisation dans SALOME 218

M.2.1 Echange d'une géométrie du composant GEOM vers SILENE . . 218

M.2.2 Application à un dispositif expérimental complexe : TANOXOS . 219

Table des figures

1.1 Représentation des déformations de pastilles combustibles et de la gaine d'un crayon, sous les effets de l'irradiation (image CEA/DEN). 35

1.2 Section horizontale du réacteur OSIRIS 37

1.3 Cœur et réflecteur du RJH (extrait de documents techniques CEA / AREVA_TA). 39

1.4 Présentation d'un assemblage combustible du RJH. 39

1.5 Représentation d'un dispositif CHOUCA contenant trois échantillons. . . 41

1.6 Représentation d'un dispositif convertisseur à 72 aiguilles. 41

1.7 Aiguilles du convertisseur en réseau triangulaire. 42

2.1 Exemples de maillages structurés, régulier à gauche et irrégulier à droite. 48

2.2 Exemple d'un maillage non structuré. 48

2.3 Maillage composé de quatre triangles et son graphe d'adjacence. Les indices locaux de chaque triangle sont entourés. 48

2.4 Représentation et exemple d'un maillage par arêtes étoilées (winged-edge). 49

2.5 Exemple d'un maillage non conforme 50

2.6 Exemple d'un maillage mixte non conforme 50

2.7 Exemple d'un maillage mixte triangle/quadrangle. 50

2.8 Exemple d'un maillage mixte contenant des bords de mailles en arc de cercle, utilisé par la méthode des caractéristiques en neutronique. 51

2.9 Section efficace microscopique totale du ^2H à gauche, et du ^{235}U (résonnante) à droite. 54

2.10 Profils de vitesse d'un fluide incompressible dans une conduite selon le régime d'écoulement. 57

2.11 Schéma de couplage explicite fort. 58

2.12 Schéma de couplage explicite faible. 59

3.1 Représentation d'une face par un modèle B-Rep. 62

3.2 Représentation B-Rep d'une face trouée. 62

3.3 Représentation B-Rep du cube. 63

3.4 Exemples d'applications des opérateurs ensemblistes union ($\cup$), intersection ($\cap$) et différence ($\setminus$) d'un cube et d'un cylindre. 65

3.5 Un arbre CSG et l'objet A qu'il modélise par l'expression $A = (B \cup C) \setminus D$. 65

3.6 Illustration de l'opération de régularisation. L'objet a est non régularisé, tandis que b est sa version régularisée. 66

3.7 Conversion d'un modèle CSG vers un modèle B-Rep. 68

3.8 Représentation d'un clou et de ses paramètres. 69

3.9 Modélisation paramétrique d'une canalisation : le rayon R du cylindre d'eau est égal au rayon interne (R_{int}) de la canalisation. 70

4.1 Présentation des principaux composants de la plate-forme SALOME. . . 75

4.2 Objets de base pour la construction d'Objets Technologiques spécifiques (diagramme UML). 79

4.3 Diagramme UML d'un assemblage de crayons (Objets Technologiques DESCARTES) 79

4.4 Assemblages pouvant être représentés par l'Objet Technologique Assembly. 80

4.5 Diagramme UML d'un crayon (Objets Technologiques DESCARTES) et de ses spécialisations 81

4.6 Eléments composant un Objet Technologique représentant un assemblage pour RNR. 81

4.7 Boite englobante et grille géométrique d'un assemblage REP décrit dans un Objet Technologique. 82

4.8 Eléments de type "crayon" (classe Rod) composant un Objet Technologique représentant un assemblage pour REP. 82

5.1 Diagramme des classes développées pour la modélisation d'un CHOUCA. 92

5.2 Éléments composant un Objet Technologique représentant un CHOUCA. 92

5.3 Rotation paramétrée d'un CHOUCA. 93

6.1 Exemple de modification des surfaces de deux objets connexes. 98

6.2 Exemple d'un dispositif d'irradiation. 99

6.3 Différentes vues possibles d'une même pièce (A : La pièce finale, B : La pièce et son outil d'usinage en vue de la fabrication, C : La pièce vue pour une modélisation). 102

6.4 Échanges de données entre le modèle d'irradiation des dispositifs (MID) et différentes simulations. 103

6.5 Principe d'échange de champ par un maillage de référence (union des maillages de chaque simulation). 104

6.6 Principe d'échange de champs sans maillage de référence. 106

7.1 Diagramme UML d'un cœur (Objets Technologiques DESCARTES) . . 110

7.2 Principe de modélisation de la géométrie d'un dispositif par un arbre d'Entités Technologiques 112
7.3 Diagramme de classe (non complet) des Entités Technologiques 113
7.4 Diagramme de classe de Problem (inspiré de [Per06]) 117
7.5 Diagramme de classe de ComputationSequence et CalculationScheme . . 118
7.6 Exemple de résultats d'un calcul neutronique sur un domaine composé de deux milieux différents. 119
7.7 Diagramme de séquence d'une séquence de simulations (deux pas de temps). 120
7.8 Diagramme de séquence d'une séquence de simulations avec des échanges de données. 121
7.9 Digaramme de classe de TransformedGeometry, PyGeomFile et GeomObjectGeometry. 123
7.10 Modélisation par Entités Technologiques d'un tube en Fer contenant de l'eau et visualisation dans SALOME. 124
7.11 Utilisation d'une séquence de calcul au travers de l'interface graphique de SALOME. 126

8.1 Présentation du dispositif TANOXOS 134
8.2 Maillage en régions de flux plat du dispositif TANOXOS 135
8.3 Présentation du dispositif TANOXOS modélisé par l'arbre d'Entités Technologiques 136
8.4 Le Foureau contient un cylindre d'eau. 136
8.5 Barillet du dispositif TANOXOS et démonstration du paramétrage du nombre de crayons. 137
8.6 Schéma simplifié du calcul neutronique (calcul d'évolution quasi sationnaire) 138
8.7 Schéma simplifié du calcul du comportement d'un crayon REP 139
8.8 Schéma du couplage neutronique / comportement combustible 140

9.1 Maillage en régions de flux plat du cœur d'OSIRIS 144
9.2 Représentation par SILENE du flux en neutrons thermiques calculé sur le cœur d'OSIRIS neuf avec le dispositif TANOXOS (unité arbitraire). . 144
9.3 Représentation par SALOME (développé au paragraphe 9.2.2) du Flux en neutrons thermiques calculé sur le cœur d'OSIRIS au cycle 192 avec le dispositif TANOXOS (unité arbitraire). 145
9.4 Schéma d'intégration du calcul neutronique 146
9.5 Exemple de modélisation d'objets géométriques SILENE 147
9.6 Représentation STEP équivalente à la figure 9.5 148
9.7 Sens de parcours et périmètre d'une maille 149
9.8 Orientation d'une maille 150

9.9 Représentation des flux en neutrons thermiques dans SALOME du RJH et du réacteur OSIRIS 152

10.1 Présentation d'une instance de Neutronic dans l'interface SALOME . . . 154

10.2 Lancement de la méthode Initialize d'une instance Neutronic et mise en attente du code APOLLO2 155

10.3 Exécution d'une séquence de calculs comprenant une simulation neutronique couplées avec six simulations de comportement combustible de crayons REP (crayons du dispositif TANOXOS). 156

A.1 Variation de la température dans un crayon combustible en fonction de la puissance linéique 169

C.1 Diagramme de classes (incomplet) du protocole d'application 214 utilisé. 176

C.2 Exemple de géométrie. 176

C.3 Périmètre de la face 2. 177

C.4 Face 1 et ses périmètres. 177

C.5 Diagramme objets des instances Advanced_face. 178

C.6 Diagramme objets de la face 2, représentant les différents types de bords. 178

C.7 Diagramme objets du bord 1(Segment) de la face 2 dans la représentation STEP AP214. 179

C.8 Diagramme objets du bord 3 (Arc) de la face 2 dans la représentation STEP AP214. 180

D.1 Diagramme UML d'une classe 183

D.2 Intégration d'un composant TUTO contenant la classe A. 185

F.1 Diagramme de classe de Circle 189

G.1 Cube unitaire et ses entités, construit par le composant GEOM. 195

H.1 Menu de géométrie 197

H.2 Menu géométrie 0D 197

H.3 Menu géometrie 1D 198

H.4 Menu géometrie 2D 198

H.5 Menu géometrie 3D 198

H.6 Menu géometrie Special 199

H.7 Menu des transformations 199

H.8 Menu des materiaux 200

H.9 Menu des Entités Technologiques 200

H.10 Menu des problèmes 200

H.11 Menu des problèmes stationnaires 200
H.12 Menu des problèmes instationnaires 200
H.13 Menu des problèmes spécifiques 201

J.1 Variation du paramètre *rayon* de l'objet *PyGeomFile*. 206

K.1 Visualisation du dispositif TANOXOS modélisé par un arbre d'Entités Technologiques. 207

L.1 Ecarts APOLLO2 − TRIPOLI4 sur les k_{eff} du calcul de cœur d'OSIRIS. 211
L.2 Distribution de puissance par quart de plaque dans le cœur d'OSIRIS calculé par T4. 212
L.3 Ecarts APOLLO2 − TRIPOLI4 sur les taux de fission par quart de plaque du cœur d'OSIRIS. 212
L.4 Répartition des écarts de puissance par quart de plaque. 213
L.5 Comparaison TRIPOLI4/APOLLO2 (tirée de [BSA+06]) des puissances calculées dans les crayons et des flux thermiques des collectrons de TANOXOS. 213

M.1 Modification de l'interface SILENE 215
M.2 Choix du fichier STEP 216
M.3 Interface proposant d'associer un fichier de milieux 216
M.4 Choix du fichier de milieux 217
M.5 Ouverture d'une géométrie STEP dans SILENE 217
M.6 Ouverture du cas école dans SILENE 218
M.7 Application au dispositif TANOXOS 219

Introduction

Une conséquence importante du premier choc pétrolier de 1973 fut le développement en France de l'énergie électronucléaire. Aujourd'hui, avec 58 tranches de réacteurs à eau pressurisée (REP), 80% de la production française d'électricité est d'origine nucléaire. Avec le retour d'expérience actuel, il est envisagé d'allonger la durée de vie des centrales et de leurs composants, initialement prévus pour fonctionner 30 ans. De plus, de nouveaux réacteurs, dits de 4^e génération sont à l'étude, pour relever les défis de la préservation de la ressource Uranium ainsi que de la minimisation des déchets nucléaires. C'est pourquoi, il est nécessaire de pouvoir tester et prédire le comportement physico-chimique sous irradiation des matériaux qui composent ces systèmes.

Pour cela, le physicien dispose de deux moyens d'investigation complémentaires, la simulation numérique et l'expérimentation. Les expériences sont souvent réalisées dans des réacteurs de recherche, de taille réduite. Ils permettent un accès facilité pour la manipulation d'échantillons sous des rayonnements neutroniques et photoniques intenses, afin d'obtenir un facteur d'accélération suffisant pour évaluer le vieillissement sous irradiation des matériaux. Ces échantillons sont introduits à l'intérieur de dispositifs expérimentaux, qui sont des systèmes mécaniques instrumentés, positionnés dans le cœur ou en périphérie de celui-ci. La simulation des phénomènes sous irradiation, de l'échelle microscopique à l'échelle macroscopique, fait appel à de grands codes de calculs, qui traitent de manière approfondie des disciplines aussi variées que la neutronique (calcul du flux de neutrons), la photonique (calcul du flux de photons), la thermohydraulique (calcul du refroidissement des matériaux), la thermomécanique (calcul des contraintes thermiques et mécaniques des matériaux) ou le comportement microscopique des matériaux (calcul des cascades de déplacement des atomes, du relâchement de produits de fission gazeux...).

En France, c'est dans le réacteur OSIRIS que sont réalisées à ce jour ces expériences d'irradiation. Construit dans les années 1960, ce réacteur est situé au centre CEA de Saclay. Ce réacteur devrait être remplacé dans la prochaine décennie par le Réacteur Jules Horowitz (RJH), en cours d'élaboration, sur le centre CEA de Cadarache. En parallèle de ces études, il est indispensable de développer des outils de simulation qui permettent d'améliorer le niveau de modélisation de ces expériences d'irradiation. Actuellement, des simulations numériques sont fréquemment réalisées sur les dispositifs d'irradiation, enchaînant les calculs par discipline, généralement sans possibilité d'effectuer un réel couplage des phénomènes physiques.

L'objectif de ce travail de thèse est de concevoir et de développer un outil de simulation fine, permettant le couplage de modèles numériques traitant des comportements

neutroniques, thermohydrauliques et thermomécaniques des dispositifs expérimentaux d'un réacteur d'irradiation technologique. Cet outil devra être le support de la conception des dispositifs, de la détermination des protocoles expérimentaux en réacteur et de l'interprétation post-irradiation de l'expérience.

Le problème exact de la modélisation d'un dispositif d'irradiation consisterait à calculer les paramètres de la neutronique, de la thermohydraulique et du comportement des matériaux dans un même système, dont les équations seraient dépendantes les unes des autres. Ce type de couplage dit *implicite* nécessite de coupler les disciplines au niveau des équations de la physique. La complexité de cette approche implique qu'on la substitue généralement par la mise en œuvre d'un couplage *explicite*, pour lequel la résolution du problème est obtenue par un enchaînement séquentiel et automatique des opérations où chaque code de calcul résout les équations d'une discipline et renseigne les données d'entrée du code suivant.

Le couplage explicite nécessite un standard d'échange des données partagées par les codes de calcul à coupler. Cette nécessité apparaît tant au niveau des données d'entrée (modélisation géométrique du système, définition des matériaux et des conditions aux limites) qu'au niveau des données de sortie (échanges de résultats sur les géométries, propriétés physico-chimiques).

Dans l'objectif de faciliter les échanges de données lors du couplage de codes de calcul, de nombreux partenaires[1], dont le CEA et EDF, se sont associés pour développer la plate-forme SALOME[2]. Cette plate-forme propose un ensemble d'outils de pré/post traitements de données, permettant de construire des modèles géométriques CAO[3], de calculer des maillages de géométries, de visualiser des données sur des maillages et de distribuer des composants au travers de réseaux informatiques. Des applications *métier* peuvent être développées et intégrées à SALOME sous la forme de composants. SALOME propose en outre un modèle pour la description de maillages et de champs de valeurs. Ce modèle est intégré dans un format d'échange de données (le format MED[4]), permettant de répondre en partie au problème d'échange de résultats.

La problématique du modèle de données commun aux disciplines a été abordée dans le contexte de la plate-forme SALOME par le groupe de travail PAL/DISCIPLINE[5]. Un modèle paramétrique de données, commun à plusieurs disciplines de physique, a été développé par cette équipe. Ce modèle, ayant pour objectif de décrire précisément un réacteur électrogène[6], est basé sur des objets appelés Objets Technologiques, en référence à la représentation par fonction technologique des composants de réacteurs. Les objets du modèle, créés et paramétrés pour une étude, sont ensuite interprétés par chaque code de calcul pour générer les données d'entrée dans le modèle de données interne du code : il s'agit d'un modèle descriptif.

[1] EDF, BUREAU VERITAS, OPEN CASCADE, PRINCIPIA R&D, CEDRAT, EADS CCR, CEA, LIP6 (Laboratoire Informatique de Paris 6), LEG (Laboratoire d'Electrotechnique de Grenoble)

[2] http ://www.salome-platform.org

[3] Conception Assistée par Ordinateur

[4] Modèle d'Echange de Données

[5] Groupe de travail CEA/EDF, PAL : Projet Architecture Logicielle

[6] de type REP (Réacteur à Eau Préssurisée) ou RNR (Réacteur à Neutrons Rapides)

La question qui se pose pour le développement de l'outil de simulation des dispositifs expérimentaux, est de savoir comment généraliser cette architecture aux cas de couplages pour les dispositifs des réacteurs d'irradiation et quels modèles devront être développés.

Pour cela, il importe de bien comprendre les spécificités des réacteurs expérimentaux. La première partie de ce document va permettre de mieux situer ce thème. Des dispositifs d'irradiation représentatifs des réacteurs OSIRIS et RJH sont présentés dans le premier chapitre. Le choix de la famille de dispositifs de référence, pour un premier couplage, s'est porté sur des dispositifs d'irradiation de crayons combustibles situés en périphérie du réacteur. L'intérêt de ce type de dispositif réside dans la quantité et la qualité des résultats expérimentaux afin d'assurer une qualification[7] ultérieure des simulations numériques réalisées dans la plate-forme.

Les méthodes de calcul neutronique, thermohydraulique et thermomécanique pour cette famille de dispositifs sont détaillées dans le deuxième chapitre afin de se familiariser avec les traitements numériques appropriés aux différentes disciplines de la physique et de référencer les données échangées entre les codes.

Connaissant le type de géométries utilisées, il s'agit ensuite d'étudier quel modèle de description géométrique est le plus adapté. Le troisième chapitre montre que certains modèles ont pour avantage d'être plus orientés *utilisateur* et sont plus faciles d'emploi, d'autres présentent des avantages en terme de simulation numérique. Pour la modélisation des dispositifs, un modèle hybride, basé sur les deux approches, alliant simplicité d'utilisation et permettant d'accéder simplement à des relations de voisinage entre objets, est retenu.

Le quatrième chapitre traite de l'environnement informatique de l'outil de simulation. Compte tenu des nombreuses fonctionnalités de la plate-forme SALOME, il convient de présenter la manière d'y intégrer de nouveaux développements. L'outil de simulation peut être conçu comme un composant logiciel de SALOME pouvant communiquer avec les composants existants. Dans ce cadre, l'exemple de l'intégration des Objets Technologiques de PAL/DISCIPLINE dans SALOME y est détaillé.

La seconde partie de ce document décrit la définition et le développement de l'architecture de l'outil de simulation. Dans le cinquième chapitre, la capacité des Objets Technologiques à décrire un dispositif d'irradiation a été évaluée. Les Objets Technologiques utilisés disposent des modèles définis au préalable, et seul un paramétrage de ce modèle est accessible à l'utilisateur. La définition de la géométrie d'étude est donc inhérente au modèle.

Pour les dispositifs, l'objectif est que l'utilisateur définisse et assemble une géométrie quelconque. Il faut donc bâtir un modèle de données permettant de construire des géométries différentes qui proposent les mêmes interfaces (mêmes attributs, mêmes méthodes) quel que soit l'objet modélisé. Tenant compte des objectifs et des différentes contraintes propres à la modélisation des dispositifs expérimentaux, le sixième chapitre s'attache à définir un tel modèle de données.

[7] Dans le sens d'une validation expérimentale

Le septième chapitre présente le développement de ce modèle de données. Ce modèle, basé sur les mêmes outils informatiques que les Objets Technologiques, a été baptisé modèle d'*Entités Technologiques*. Les Entités Technologiques permettent de décrire un dispositif d'irradiation à travers un arbre de construction. Chaque nœud de cet arbre représente un objet sous la forme d'une association géométrie/matériau. Cette approche est appliquée au modèle géométrique hybride présenté au chapitre 3. A partir de cet arbre, un modèle géométrique de CAO peut être automatiquement construit en faisant appel aux services du composant GEOM[8] de SALOME. Ce modèle paramétrique peut ainsi être mis à jour dynamiquement et visualisé dans SALOME. Les informations contenues dans les Entités Technologiques peuvent être lues et modifiées lors des simulations des phénomènes physiques. Pour cela, les modélisations sont encapsulées dans une interface spécifique, fournissant des méthodes d'accés aux données décrites par les Entités Technologiques.

La troisième partie de ce document concerne la mise en œuvre des développements sur un exemple de couplage neutronique et comportement du combustible sur le dispositif TANOXOS du réacteur OSIRIS. Il s'agit d'un dispositif contenant six crayons combustibles de poudre d'oxyde d'Uranium ou d'oxyde mixte Uranium-Plutonium, irradié en périphérie nord du cœur et dont l'objectif est l'étude du comportement du combustible (relâchement des gaz de fission) à fort taux de combustion. Cette application est présentée dans le huitième chapitre, et sera détaillée dans les chapitres suivants. L'objectif est d'obtenir la distribution de température dans les crayons combustibles.

Il s'agit en premier lieu de réaliser la modélisation neutronique du dispositif ; cette modélisation, détaillée dans le neuvième chapitre, est intégrée à un schéma de calcul neutronique appliqué au cœur complet du réacteur OSIRIS et basé sur le code APOLLO2 (calcul du transport des neutrons à 2D à l'aide de la méthode des caractéristiques). L'intérêt de ce schéma est double : il permet de traiter la géométrie générale du dispositif et de le positionner à n'importe quel endroit du cœur avec un schéma de calcul unique. En contrepartie, la principale difficulté est d'adapter la géométrie créée par les Entités Technologiques au modèle APOLLO2, et de la même manière, traduire les données calculées dans le format MED. Le calcul neutronique fournit la puissance linéique générée par le dispositif et introduite dans les calculs de comportement combustible.

La mise en œuvre du séquencement automatique des calculs, réalisé avec les outils développés durant la thèse, est détaillée dans le dixième chapitre.

Le bilan de cette application permettra d'évaluer l'apport de l'architecture sur les gains réalisés en terme de mise en œuvre des simulations numériques, ainsi que sur les possibilités d'adaptation à d'autres familles de dispositifs.

[8] Composant proposant des services de calcul de géométries

Première partie

Contexte et enjeux de la thèse

Chapitre 1

Irradiations dans les réacteurs expérimentaux

Toute source d'irradiation induit des modifications sur les matériaux qui induisent de fortes conséquences sur leurs comportements physiques. Dans ce chapitre, on s'intéresse aux effets de l'irradiation sur les matériaux présents dans les cœurs de Réacteurs à Eau Pressurisée (REP) ou à caloporteur sodium (Réacteurs à Neutrons Rapides RNR), en détaillant les conséquences de cette irradiation sur les matériaux de structure ainsi que sur les combustibles nucléaires.

Ce chapitre peut être jugé trop schématique pour un spécialiste de physique des réacteurs, et beaucoup trop complexe pour un informaticien. Son objectif est d'essayer de bien positionner le cadre de ce travail de thèse.

1.1 Comportement des matériaux sous irradiation

1.1.1 Matériaux de structure

Les éléments de structure qui supportent l'ensemble des composants du réacteur et assurent le confinement du caloporteur et/ou du modérateur doivent résister aux multiples rayonnements provenant du cœur. Les alliages industriels concernent essentiellement les aciers, comme les aciers de construction pour les cuves de réacteurs, ou les aciers inoxydables pour les structures internes des réacteurs thermiques ou les éléments de structure des réacteurs à neutrons rapides. Ces métaux sont tous soumis à un dommage d'irradiation lié aux déplacements d'atomes produits par le flux de neutrons rapides (neutrons dont l'énergie est supérieure à 907 keV).

Pour les aciers inoxydables austénitiques[1] soumis à de très fortes fluences neutroniques, on observe un phénomène de gonflement.
Il se traduit par une diminution de la densité, et donc par une augmentation des dimensions des éléments de structure. Pour ces matériaux, le gonflement est la limitation principale de l'utilisation dans les réacteurs à neutrons rapides. Une des contraintes

[1] Acier dont la structure cristalline est de type cubique faces centrées obtenue par alliage avec des éléments de type Ni, Mn, N ou Cu

majeures induite par ce gonflement, en plus de la distorsion significative des aiguilles combustibles de RNR, est le gerbage des tubes hexagonaux formant les assemblages de ces réacteurs.
Pour les assemblages situés en périphérie du cœur du réacteur, il existe un gradient de flux neutronique significatif entre les faces du tube hexagonal.
La face située plus près du centre reçoit un flux neutronique plus important : elle sera donc le siège d'un gonflement plus précoce et plus intense que la face opposée. Cette déformation différentielle se traduit par une flexion du tube qui a tendance à ouvrir les assemblages vers l'extérieur, ce qui peut entraîner des difficultés de manutention au rechargement des assemblages.
Une classe d'aciers prometteurs vis-à-vis du gonflement rassemble les aciers essentiellement ferritiques[2], martensitiques[3] et duplex[4], de structure cristallographique cubique centrée présentant moins de gonflement que les structures cristallographiques cubiques faces centrées. Les raisons de ces différences restent encore ignorées et font l'objet de travaux de recherche analytique [Lem04].

Le dommage d'irradiation subi par les cuves en acier de construction des REP se traduit principalement par l'augmentation de la température de transition ductile-fragile[5]. Ce phénomène conduit à une fragilisation sous irradiation des aciers de cuve, il est essentiellement contrôlé par trois paramètres : la dose reçue par la cuve, la température de l'irradiation et la composition de l'acier irradié. Compte tenu des faibles fluences reçues par la cuve ($\phi_{max} < 10^{20}$ n.cm^{-2}), l'augmentation des propriétés mécaniques reste toujours faible, allant d'un effet non mesurable à une augmentation de 5 à 10% de la limite d'élasticité σ_E. Cependant, cette évolution est la trace du développement de modifications structurales affectant les mécanismes de déformation et justifie un programme important de surveillance des cuves des REP, qui consiste à contrôler régulièrement l'évolution de la fragilité d'éprouvettes témoins subissant le même historique d'irradiation que la cuve elle même.

Le cas de la gaine en zirconium des REP est traité dans le paragraphe suivant.

1.1.2 Combustible nucléaire

Durant leurs séjours en réacteur, les combustibles sont le siège de nombreux phénomènes physiques d'origines neutronique, thermique, mécanique et physico-chimique qui sont

[2] Les alliages acier-chrome favorisent la forme ferritique (la structure cristallographique du fer est alors cubique à corps centré).

[3] Un acier martensitique à une forte teneur en carbone comparée aux aciers austénitiques (dont la structure cristallographique est cubique à faces centrées).

[4] les aciers dit duplex sont des aciers austéno-ferritiques.

[5] Température pour laquelle le matériau passe de la zone de rupture fragile (basses températures) à la zone de rupture ductile (températures plus élevées). Au dessous de la température de transition, le métal fragile est incapable d'accommoder les concentrations de contraintes au fond d'une fissure. Une faible énergie telle que celle mise en jeu par des vibrations ou des chocs, suffit à propager une rupture, à partir d'un défaut de surface. Au dessus de la température de transition, il faut une énergie importante pour ouvrir une micro-fissure préexistante au fond de laquelle il n'y a pas de concentration de contraintes. Par conséquent, les amorces de rupture correspondant aux défauts de surface ne se propagent pas. Sous l'effet de l'irradiation, la formation de défauts ponctuels par les dommages conduit à une augmentation de la résistance mécanique et le matériau perd de sa ductilité. Cela conduit à une augmentation de la température de transition ductile-fragile, et le matériau devient fragile à plus haute température.

totalement interdépendants. Ces phénomènes engendrent une importante transformation des combustibles au cours de leur irradiation.

Dans le cas du combustible REP actuel, on assiste à une demande des exploitants de centrale pour obtenir des combustibles permettant d'atteindre des taux de combustion (l'énergie maximale que l'on pourra tirer d'un assemblage) plus importants. L'électricien souhaite également pouvoir améliorer la manœuvrabilité de ses tranches nucléaires. En France, les 3/4 de la production électrique étant assurée par le nucléaire, le parc peut être amené à fonctionner selon des régimes différents de 100% de la puissance nominale (suivi de charge, suivi de réseau, fonctionnement prolongé à puissance intermédiaire).
La plupart du temps, ces variations de puissance se traduisent par des variations de contraintes entre la pastille combustible (UO_2 ou poudre d'oxyde mixte UO_2, PuO_2) et la gaine en zircaloy : c'est l'interaction pastille-gaine. Le dimensionnement du crayon combustible REP s'intéresse donc à toutes les sources de sollicitations possibles sur la gaine susceptibles d'engendrer une rupture de cette première barrière.

Il est nécessaire de comprendre et modéliser tous les phénomènes à l'origine de ces sollicitations potentielles aussi bien en fonctionnement nominal qu'en conditions accidentelles.

On distingue principalement 3 phénomènes qui limitent actuellement le taux de combustion maximal que l'on peut atteindre avec les assemblages installés dans les réacteurs actuels :

- Le relâchement des gaz de fission augmentant la pression interne dans les crayons REP avec l'augmentation du taux de combustion ;
- La corrosion du gainage ;
- L'Interaction Pastille Gaine (IPG).

Relâchement des gaz de fission et pression interne dans les crayons REP

La pression interne dans les crayons REP à fort taux de combustion doit rester en deçà d'une valeur qui serait capable, par fluage[6] de la gaine, de réouvrir un jeu combustible/gaine. En effet, l'occurrence d'un tel phénomène entraînerait une augmentation des températures du combustible par la dégradation brutale du coefficient d'échange caloporteur/combustible. Ce qui accroîtrait le relâchement de gaz de fissions supplémentaires d'où un risque d'emballement du phénomène.
Tant que la pression dans le crayon reste inférieure à la pression du caloporteur (155 bars), ce risque n'existe pas. La pression interne peut même devenir légèrement supérieure à celle du caloporteur, car le gonflement du combustible peut compenser le fluage de la gaine et maintenir le jeu fermé.
On ne tolère cependant qu'un faible dépassement de la pression du caloporteur, et les crayons sont dimensionnés pour que la pression interne ne dépasse pas une valeur d'environ 180 bars, valeur qui peut être revue à la hausse si on utilise un matériau de gainage avec une plus faible vitesse de fluage tel que le M5 (Paragraphe suivant).

[6] Soumis à une contrainte de température et d'irradiation, un métal se déforme de manière régulière : on parle alors de fluage.

Pour diminuer les pressions internes, on peut jouer sur 3 paramètres : la pression initiale d'hélium, le volume libre et la capacité de rétention des gaz de fission du combustible. C'est pourquoi pour augmenter les taux de combustion, la pression initiale de gonflage a été abaissée à 25 à 15 bars, mais il parait difficile de descendre en dessous de cette valeur.
Pour augmenter le volume libre des crayons, il est possible d'enlever un certain nombre de pastilles, mais ce nombre est limité par la puissance que l'on veut dégager pour un assemblage donné (la hauteur de l'assemblage étant fixée à la conception du cœur).
Enfin, le troisième mode d'action pour réduire les pressions de fin de vie est de développer un combustible ayant une capacité accrue de rétention des gaz de fission. Un vaste programme a été lancé sur des pastilles en UO_2 dopé au chrome.

Ces combustibles ont d'abord été testés en réacteur expérimental puis dans des crayons REP. Les résultats de ces campagnes permettront de constituer le dossier de Sûreté pour commencer à charger un cœur complet avec ce produit à partir de 2012. Des études similaires sont en cours au stade amont sur des combustibles MOX. Des irradiations en réacteur expérimental sont en cours pour évaluer le comportement des microstructures de ce combustible.

Corrosion du gainage

Dans les REP actuels, l'alliage le plus utilisé en France pour les tubes de gaine est le Zircalloy 4 (Zy4), contenant 1.2 % à 1.7 % d'étain, du fer, du chrome et de l'oxygène.
Ce gainage ne permet pas de dépasser 5 cycles d'un an en REP français à cause de son oxydation et son hydruration. La couche d'oxydation après 5 ans d'utilisation peut atteindre près de 80 à 100 μm à l'étage le plus chaud du combustible, ce qui représente parfois 1/6 de l'épaisseur totale de la gaine. Par phénomène de desquamation, l'oxyde peut partir de la gaine et la corrosion peut continuer à se développer. L'hydruration du zirconium (formation de ZrH_2), lorsqu'elle est importante, dégrade les propriétés mécaniques de la gaine.
Les quatre facteurs principaux influençant la corrosion sont la chimie de l'eau primaire (pH légèrement basique pour compenser l'addition d'acide borique à des fins de contrôle de la criticité du réacteur), la température de l'eau, le flux thermique (gradient de température sur la gaine), le flux nucléaire.
De nombreux alliages ont été développés, ayant en commun la réduction de la teneur en étain pour limiter la corrosion. L'alliage retenu doit posséder un meilleur comportement à la corrosion, sans perdre ses caractéristiques mécaniques. Les réacteurs expérimentaux sont des outils de première importance pour tester ces nouveaux gainages.

L'interaction pastille gaine (IPG)

Au cours de la vie en réacteur, diverses forces s'appliquent sur la pastille et sur la gaine :

- La pastille, obtenue par frittage, commence d'abord par se densifier. Puis, sous l'action notamment du relâchement des produits de fission gazeux, a tendance à se dilater. La déformation de la pastille n'est pas isotrope, mais prend la forme de diabolo (voir figure 1.1). C'est pour cela que la pastille possède deux dishing [7] et des chanfreins en

[7] Sur ses faces supérieures et inférieures, la pastille n'est pas parfaitement plane, mais possède un évidement

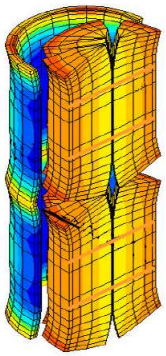


FIG. 1.1 – Représentation des déformations de pastilles combustibles et de la gaine d'un crayon, sous les effets de l'irradiation (image CEA/DEN).

fabrication pour limiter ce phénomène ;
- La gaine flue sous la pression du réfrigérant externe et le jeu entre la pastille et la gaine décroît progressivement, jusqu'à contact complet vers 2 ou 3 cycles. Ensuite, la pastille pousse sur la gaine qui va fluer dans l'autre sens.

Ces phénomènes se produisent en conditions nominales. Mais si le réacteur est soumis à un accroissement de puissance rapide, ils seront exacerbés. L'augmentation de la puissance linéique exprimée en W/cm au cœur du crayon se traduit également par une dilatation plus importante. Enfin, parmi les produits de fission relâchés, la présence d'iode peut entraîner une corrosion sous contrainte de la gaine, pouvant aller jusqu'à sa rupture éventuelle. La corrosion sous contrainte se traduit par l'apparition de fissures aux endroits de contraintes maximales (face aux inter-pastilles par exemple).
L'électricien souhaite pouvoir faire fonctionner son réacteur avec le maximum de souplesse, ce qui a pour conséquence de faire varier rapidement sa puissance linéique.
L'IPG l'oblige à respecter des puissances maximales, et à limiter la vitesse de ces variations. Dans la pratique, les critères de fonctionnement sont basés sur des expériences dans un réacteur d'irradiation technologique comme OSIRIS, au cours desquelles on soumet un petit crayon combustible, préalablement conditionné à une puissance linéique normale, à une augmentation rapide de celle-ci (typiquement, de l'ordre de 100W/cm/-min), de façon à déterminer un seuil de rupture. Ces essais sont appelés rampes de puissance.

Après la description des principaux phénomènes physiques qui interviennent dans les cœurs des réacteurs nucléaires, la suite de ce chapitre présente les réacteurs qui ont été au centre de cette étude.

1.2 Réacteur OSIRIS

Le réacteur OSIRIS est un réacteur expérimental d'une puissance thermique de 70 MW implanté au centre CEA Saclay.

1.2.1 Description du cœur

OSIRIS est un réacteur de type piscine à eau légère et à cœur ouvert ($57x57x60cm^3$), dont le but principal est d'effectuer des essais et d'irradier sous haut flux de neutrons des éléments combustibles et des matériaux de structure des centrales électronucléaires de puissance ainsi que de produire des radioéléments pour l'industrie et la médecine.

Ce principe de conception d'un réacteur piscine et à cœur ouvert permet l'accès direct au cœur, facilité par l'absence de cuve de pressurisation. Pour créer une pressurisation dynamique, on a adopté un sens de circulation ascendant pour l'eau de réfrigération de combustible à l'intérieur du cœur.
Une cheminée surmonte le cœur, elle est destinée à éviter les remontées d'eau fortement activée dans la piscine. Elle limite le mélange entre l'eau sortant du cœur et l'eau de la piscine.

Un caisson en Zircalloy de section rectangulaire de 4 cm d'épaisseur entoure le cœur. La présence du caisson est imposée par le sens de circulation ascendant de l'eau de refroidissement du cœur afin de positionner les éléments placés à l'intérieur de ce dernier. Le caisson constitue également un écran au rayonnement photonique pour les dispositifs expérimentaux placés à l'extérieur du cœur. Un casier alvéolé est placé à l'intérieur du cœur et comporte 8x7 emplacements carrés (pas 8,74 cm) occupés par (voir figure 1.2) :

- 38 éléments combustibles standard ;
- 6 éléments de commande ;
- 7 éléments réflecteurs en béryllium (situés sur la face sud du cœur) qui peuvent recevoir des dispositifs expérimentaux dans un trou central ;
- Les 5 autres emplacements peuvent être occupés par des dispositifs expérimentaux, boîtes à eau de géométrie extérieure identique à celle des éléments combustibles.

Depuis 1994, le réacteur OSIRIS utilise un combustible dit siliciure constitué par alliage $U_3Si_2 - Al$. L'uranium est enrichi à 19,75 % en isotope ^{235}U et répond ainsi aux critères de non prolifération correspondant au programme R.E.R.T.R[8]. Un élément combustible standard est constitué de 22 plaques d'épaisseur $0.51mm$ et gainées par de l'Aluminium d'épaisseur $0.38mm$. Les deux plaques de rive sont chargées par du poison consommable (Bore). Un élément de commande est composé de deux parties : une partie haute absorbante (hafnium) et une partie basse fissile appelée *suiveur*. Cette dernière comprend 17 plaques de constitution identique aux plaques équipant l'élément standard.

Deux barres ($n°2$ et 5), dites de sécurité, assurent l'arrêt d'urgence du réacteur, les quatre autres barres servant les unes après les autres (souvent dans l'ordre 1, 6, 3, 4) de barres de compensation et de barre de pilotage. Une seule barre est manœuvrée à la fois.

Tous les éléments placés dans le cœur (éléments combustibles, éléments de commande, éléments réflecteurs et dispositifs expérimentaux) sont liés à leur partie inférieure par une tige traversant le fond de la piscine. Le verrouillage se fait par simple rotation commandée à partir de la salle des mécanismes située sous la piscine. Le but du verrouillage des éléments est d'assurer leur blocage contre la force d'envol provoquée par le sens

[8] R.E.R.T.R. : Reduced Enrichment for Research and Test Reactors

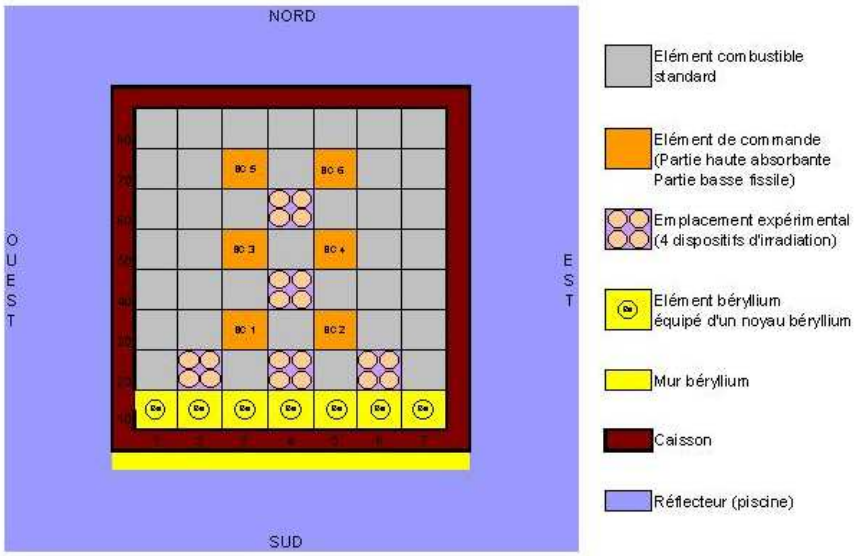


FIG. 1.2 – Section horizontale du réacteur OSIRIS

de circulation ascendant de l'eau de réfrigération à l'intérieur du cœur. Une sécurité supplémentaire est prévue sur les éléments combustibles : elle est constituée par un verrouillage sur la partie supérieure dans le casier alvéolé.

On distingue deux catégories de dispositifs : les dispositifs en cœur et les dispositifs en réflecteur.
Les premiers sont généralement utilisés pour des irradiations sous fort flux en neutrons rapides, tandis que les seconds permettent des irradiations sous fort flux en neutrons thermiques (neutrons d'énergie cinétique inférieure à 0.625 eV). Dans ces deux catégories, on peut encore séparer les dispositifs qui permettent d'irradier des combustibles et ceux qui permettent d'irradier des matériaux. Quelques dispositifs importants d'OSIRIS et qui seront sans doute présents dans le RJH sont listés ci-dessous.

1.2.2 Dispositifs en cœur

Le principal dispositif d'irradiation de matériaux en cœur est le dispositif CHOUCA. Il s'agit d'une capsule comprenant une double enveloppe constituant une barrière thermique, 6 éléments chauffants régulés en température et un porte-échantillons adapté pour chaque charge expérimentale.

La température peut atteindre des valeurs de l'ordre de 350 ° C sur les éprouvettes. Les échantillons peuvent être pressurisés ou pré-contraints et sont plongés dans du NaK (56 % Potassium, 44 % Sodium) afin d'avoir une bonne diffusion de la chaleur vers l'extérieur.

Les matériaux irradiés sont par exemple des alliages de zirconium (gainage des REP) ou des alliages Zr/Nb constituant les tubes de force de la nouvelle génération de la filière canadienne (CANDU). Parmi les dispositifs d'irradiation de combustibles en cœur, on peut citer le dispositif IRIS, destiné à irradier des plaques combustibles expérimentales

pour réacteurs de recherche. Les expériences servent à qualifier de nouveaux types de combustibles et, en particulier, leur tenue à l'irradiation en fonction du taux de combustion. Les plaques sont déchargées après chaque cycle pour réaliser des métrologies à l'aide d'un banc de mesure immergé.

1.2.3 Dispositifs en réflecteur

Les dispositifs matériaux en réflecteur sont essentiellement dédiés à l'irradiation de grandes quantités d'éprouvettes sous milieu inerte pour évaluer les caractéristiques mécaniques des matériaux (ténacité[9], décalage de température de transition ductile/fragile) après irradiation. Les contraintes pour ce type d'expériences sont de bien respecter les températures sur l'ensemble des éprouvettes (valeur de la température et stabilité au cours du temps), et la fluence visée.

Les dispositifs combustibles en réflecteur permettent principalement d'irradier des crayons combustibles dans des conditions le plus proches possibles de celles rencontrées dans les réacteurs de puissance. Les dispositifs de type ISABELLE sont particulièrement adaptés à la réalisation de rampes de puissance pour des combustibles neufs ou irradiés (étude de l'IPG), les dispositifs GRIFFONOS ont pour objectifs de réaliser des expériences analytiques (mesure de la température centrale du combustible en fonction de la puissance et du taux de combustion, étude du fluage des gaines sous flux, de la corrosion de la gaine...) sur du combustible de REP pour lesquels la vitesse du fluide caloporteur autour du crayon combustible importe peu. Les dispositifs TANOXOS sont développés pour réaliser des études sur des combustibles à microstructures avancées.

1.3 Réacteur Jules Horowitz

Le Réacteur Jules Horowitz est le remplaçant du réacteur OSIRIS, dont la divergence[10] est prévue sur le centre de Cadarache en 2014.

1.3.1 Description du cœur

D'une puissance thermique de $100MW$, il permettra des irradiations neutroniques sous des flux atteignant $10^{14} n.cm^{-2}.s^{-1}$ avec un grand nombre d'emplacements disponibles. Compte tenu des performances atteintes, une légère pressurisation du cœur (de l'ordre de $10bar$) est prévue (c'est le rôle de la virole cylindrique, en Aluminium, qui entoure le cœur, présente sur la figure 1.3).

Le cœur est de forme cylindrique, ce qui permet de minimiser les fuites de neutrons et donc d'optimiser la gestion du combustible. Il est composé de 34 éléments combustible pouvant accueillir soit des barres de commandes en Hafnium, soit des suiveurs en aluminium, soit des dispositifs expérimentaux de type capsule CHOUCA (voir paragraphe 1.3.2). Trois emplacements de grand diamètre sont réservés à l'irradiation de dispositifs

[9] La ténacité caractérise la résistance à la rupture brutale du matériau en présence d'un défaut
[10] Dans ce contexte, le terme *divergence* signifie *démarrage*.

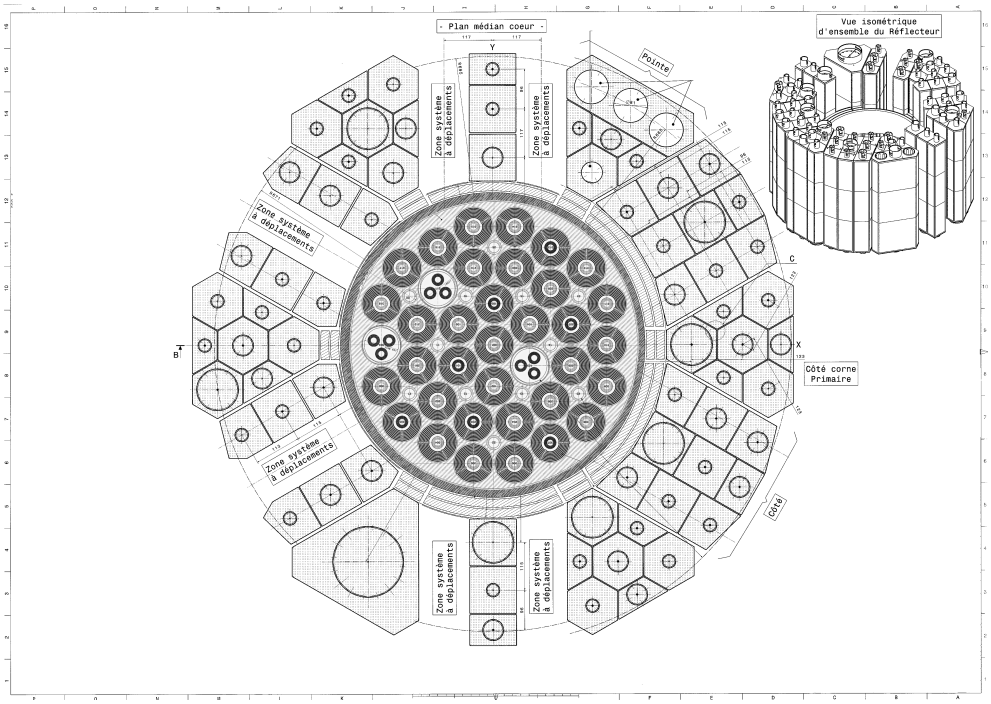


FIG. 1.3 – Cœur et réflecteur du RJH (extrait de documents techniques CEA / AREVA_TA).

CHOUCA groupés par trois, en boucles[11] spécifiques. Le pas du réseau est irrégulier, favorisant l'implantation de mandrins inter-éléments combustible pouvant également accueillir des dispositifs expérimentaux, ou des poisons consommables pour le contrôle de l'excédent de réactivité du cœur en début de vie.

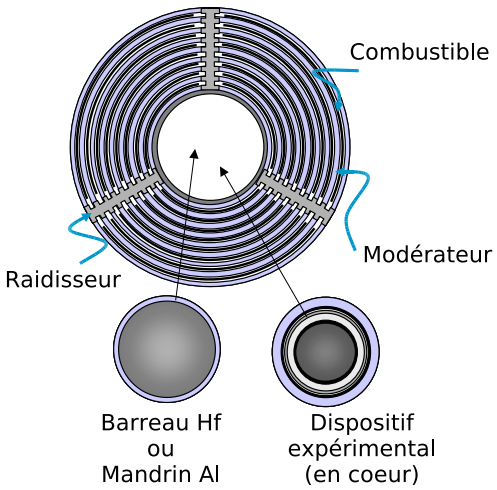


FIG. 1.4 – Présentation d'un assemblage combustible du RJH.

[11] Une boucle est un dispositif dont le circuit de refroidissement est séparé du circuit cœur, pour permettre l'utilisation d'autre caloporteur ou pour irradier des échantillons dans des conditions thermohydrauliques différentes de celles du cœur.

Un élément combustible standard est composé de trois secteurs de huit plaques combustibles serties dans des raidisseurs en aluminium (figure 1.4).

Ces plaques contiennent de l'$U_3Si_2 - Al$ enrichi à 27% de ^{235}U pour les premiers cœurs, puis de l'$UMo-Al$ à 20% d'enrichissement dès lors que la qualification de ce combustible sous irradiation sera acquise. L'âme combustible a une épaisseur de $0,61mm$, gainée par de l'$AlFeNi$[12] d'une épaisseur de $0,38mm$. La circulation d'eau entre les plaques devrait s'effectuer à une vitesse proche de $18m.s^{-1}$, ce qui entraîne des contraintes mécaniques fortes sur l'assemblage et sur les tolérances du canal entre les plaques, dont l'épaisseur est fixée à $1,95mm$.

Le cœur est entouré d'un écran en zircaloy[13] lamellé, pour protéger les expériences d'irradiation des échauffements gamma. Le réflecteur est constitué de blocs en beryllium positionnés de manière modulaire pour pouvoir disposer d'un maximum d'emplacement pour les irradiations en réflecteur. L'intérêt du beryllium est d'obtenir des flux de neutrons thermiques élevés pour les dispositifs, ainsi que thermaliser[14] sans trop absorber les neutrons, pour qu'un grand nombre d'entre eux retournent fissionner des noyaux d'Uranium dans le cœur. D'où une augmentation de la durée du cycle par une diminution des fuites de neutrons par rapport à un réflecteur en eau. Seuls certains emplacements seront dépourvus de beryllium pour pouvoir réaliser des expériences de rampe de puissance necessitant un déplacement du dispositif dans le réflecteur.

1.3.2 Dispositifs en cœur

Description du dispositif CHOUCA

Un dispositif CHOUCA est une capsule comprenant une double enveloppe qui constitue une barrière thermique, 6 éléments chauffants régulés en température et un porte échantillons adapté pour chaque charge expérimentale. Ce dispositif se situe soit au centre d'un assemblage, soit groupé par trois (dans le RJH) ou quatre (dans le réacteur OSIRIS) dans un assemblage spécifique ne contenant pas de matière fissile.

Le CHOUCA modélisé dans cette étude est représenté par la figure 1.5. Il contient trois échantillons qui baignent dans du NaK. Le NaK est un métal liquide à température ambiante. Il est isolé des milieux du réacteur par une interface de matériaux tubulaires. Cette interface permet d'une part de limiter les gradients thermiques et de ne pas dépasser les températures maximales préconisées et d'autre part d'isoler le NaK de l'eau du réacteur (mélange explosif). Des fours en périphérie du dispositif (élément chauffant), permettent de réguler la température de manière à ce que celle-ci soit la plus constante possible tout au long de l'expérience d'irradiation. Il se trouve un espace rempli d'Helium (ou bien d'azote ou de Néon) entre le tube périphérique en Aluminium et celui en Zircone. Cet espace joue un rôle de barrière thermique. Les expériences d'irradiation réalisées avec ce dispositif permettent par exemple de qualifier les alliages de zirconium utilisés pour les gaines de crayons combustibles de type REP.

[12] Alliage de l'aluminium : 1% de fer, 1% de Nickel et 1% de Magnésium.
[13] Le zircaloy est un groupe d'alliage du zirconium.
[14] C'est à dire ralentir les neutrons jusqu'au domaine d'énergie thermique.

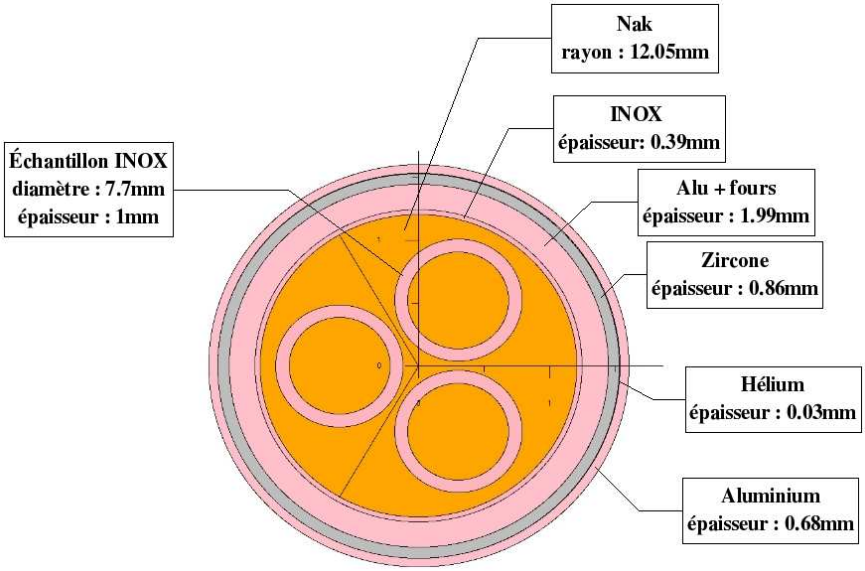


FIG. 1.5 – Représentation d'un dispositif CHOUCA contenant trois échantillons.

Description d'un dispositif de type convertisseur

L'objectif du convertisseur est d'augmenter le flux en neutrons rapides, localement, au centre du cœur RJH (assemblage central). Un échantillon sera positionné à l'intérieur de ce flux et recevra alors une forte dose en flux de neutrons rapides. Le principe de fonctionnement est de convertir le flux de neutrons thermiques en flux de neutrons rapides en augmentant le nombre de noyaux fissiles autour de l'échantillon à irradier.

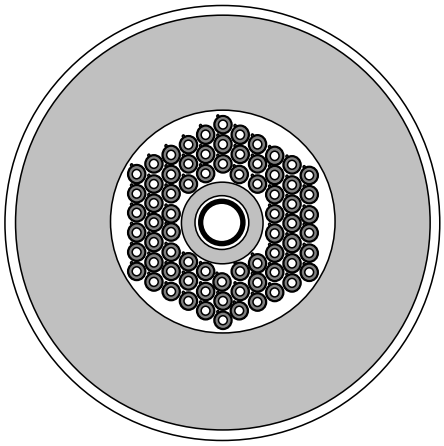


FIG. 1.6 – Représentation d'un dispositif convertisseur à 72 aiguilles.

Le dispositif convertisseur est cylindrique. Il occupe l'alvéole centrale du cœur. Le convertisseur est composé d'un tube central en aluminium, contenant de l'hélium. Dans ce tube est inséré un échantillon tubulaire en acier inoxydable. Le tube se trouve entouré d'un réseau de 72 aiguilles à combustible disposées en couronne. L'ensemble est introduit dans un tube en aluminium. Le dispositif baigne dans l'eau du circuit primaire (figure 1.6).

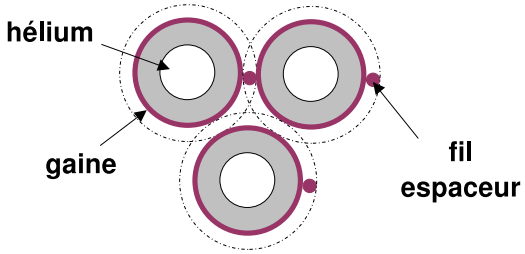


FIG. 1.7 – Aiguilles du convertisseur en réseau triangulaire.

Les aiguilles sont composées de pastilles de combustible creuses en leur centre. Le centre est rempli d'hélium. Les pastilles sont gainées en zircaloy. Les aiguilles sont disposées en réseau triangulaire, suivant un hexagone. Elles sont maintenues espacées par un fil en zircaloy enroulé hélicoïdalement autour de la gaine (figure 1.7).

Après un aperçu sur les réacteurs expérimentaux et les phénomènes physiques qui interviennent lors d'expériences d'irradiation, le chapitre suivant présente les outils de simulations utilisés par les physiciens, dans différentes disciplines de physique.

Chapitre 2

La simulation multidisciplinaire

L'irradiation d'un dispositif expérimental fait intervenir plusieurs phénomènes physiques, interdépendants, étudiés par de nombreux champs de la physique dont la neutronique et la thermohydraulique. Les études des comportements des neutrons et des écoulements des fluides au sein d'un dispositif sont deux des principales disciplines traitées dans ce contexte précis. Elles font appel à des outils de calculs numériques : les codes de calcul.

Les principaux maillages de géométries et les méthodes de calcul utilisés dans ces outils sont présentés au paragraphe 2.1. Ensuite, aux paragraphes 2.2 et 2.3, des notions de neutronique et de thermohydraulique permettront de mieux comprendre l'intérêt physique de réaliser des simulations couplées. Nous verrons dans le paragraphe 2.4 que ces simulations multiphysiques peuvent être mises en œuvre selon différents schémas de couplage.

2.1 Maillages et codes de calcul

Les codes de calcul permettent en général de traiter un grand nombre de situations rencontrées, dans différents types de réacteurs, de manière spécialisée. Par exemple, le code APOLLO2, peut aussi bien modéliser un réacteur de recherche qu'un réacteur électrogène tel qu'un REP.

La modélisation d'un problème de physique est réalisée puis paramétrée dans un jeu de données[1]. Il s'agit d'un modèle[2] spécifique au code de calcul, dans lequel est décrit a minima :

- le domaine géométrique à étudier et une discrétisation associée ;
- des caractéristiques sur des matériaux associés à cette géométrie (composition isotopique, caractéristiques mécaniques et thermiques, ...) ;
- une méthode de résolution. Les codes actuels sont modulaires et se présentent aux

[1] On parle aussi de mise en données.

[2] On peut voir apparaître aussi bien de simples fichiers de configuration de variables que de véritables programmes écrits dans des langages spécifiques à un code de calcul. Par exemple le langage GIBIANE des codes APOLLO2 et CRONOS2.

physiciens comme une *boite à outils* permettant de résoudre plusieurs types de problèmes ;
- des hypothèses sur les modèles physiques (simplifications et conditions aux limites) ;
- des requêtes pour obtenir les résultats de calcul sur tout ou partie du domaine d'étude.

Le fichier informatique contenant le jeu de données est souvent le seul moyen de dialogue avec le code de calcul. Le format d'un jeu de données et les formats de sortie sont généralement spécifiques à un code de calcul. De ce fait l'échange de données d'un code à l'autre nécessite presque toujours des développements.

Le paragraphe suivant présente les différentes méthodes de résolution rencontrées au cours de la thèse.

2.1.1 Méthodes de résolution de problèmes de physique

Ces outils de calcul sont classés dans deux grandes catégories : les méthodes de résolution déterministes et les méthodes de Monte-Carlo (méthodes de résolution statistique).

Méthodes déterministes

Les méthodes déterministes consistent en une résolution analytique des équations de la physique. Le problème de physique est posé sous forme de systèmes d'équations et le jeu de données permet de spécifier chacun des termes de ces équations.
Les équations de physique se ramènent souvent à des équations aux dérivées partielles (EDP). De ce fait, les méthodes de différences finies [Spi02b],[Spi02c] et d'éléments finis [Spi03], [Spi02d], [Spi02a], [Zie73] sont souvent utilisées pour résoudre ce type de problème.

D'une manière générale les méthodes de résolution déterministes supposent une discrétisation de la géométrie que l'on appelle maillage (voir paragraphe 2.1.2). Ce maillage peut contenir un très grand nombre d'éléments selon la précision de modélisation choisie. Du point de vue du physicien, la fonction de discrétisation du maillage peut être considérée comme étant une hypothèse simplificatrice du calcul : il considère que la variation des paramètres physiques recherchés dans une maille est connue, c'est-à-dire qu'il n'y a pas de variation ou bien une variation proche d'un polynôme particulier (Lagrange, Hermite) [Spi02d]. Il appartient donc au physicien d'élaborer un maillage de la géométrie cohérent du point de vue de la discipline de physique, de la méthode de résolution et donc du code de calcul.

Le point commun des codes déterministes est qu'ils nécessitent de disposer d'un maillage de la géométrie se rapprochant d'un graphe, ou par analogie à la Géométrie, d'un modèle B-REP [Hof89] (voir paragraphe 3.1). Ces structures de données ont l'avantage de permettre d'obtenir facilement des informations sur le voisinage d'un élément. L'intérêt d'utiliser ces maillages provient du fait que ces méthodes résolvent des systèmes d'équations aux dérivées partielles et que pour calculer des termes différentiels, il est nécessaire d'avoir une connaissance sur le voisinage des éléments.

Au CEA, les principales disciplines utilisées pour la modélisation des réacteurs nucléaires disposent de leurs codes déterministes :

- en neutronique, le code CRONOS2 [LLFM90] permet de résoudre l'équation de Boltzmann (voir paragraphe 2.2.1) en théorie du transport ou de la diffusion des neutrons ;
- en thermohydraulique, le code FLICA4 [TGR97] permet d'analyser des cœurs de réacteurs en régime permanent ou en régime transitoire incidentel, et le code CFD (Computational Fluid Dynamics) TRIO_U [KSG00] permet de réaliser des calculs mono et diphasiques en RANS (Reynolds-Averaged Navier-Stokes) , LES (Large Eddy Simulation) ou DNS (Direct Numerical Simulation).
- en thermomécanique, le code CAST3M [Fic98] permet de calculer les champs de température et de contrainte d'un matériau soumis à des sollicitations thermiques et mécaniques ;

Ces codes résolvent les équations de la physique, en général discrétisées selon les méthodes des différences finies, des volumes ou des éléments finis. En neutronique, les physiciens peuvent disposer de la méthode des caractéristiques [AdS+04] et [Roy98] récemment développée dans les codes APOLLO2 [LSC+99] et DRAGON [MRH94] pour la résolution de l'équation de transport des neutrons.

Méthode de Monte-Carlo

Les méthodes de Monte-Carlo n'effectuent généralement pas de résolution analytique d'un problème, et il n'est pas nécessaire que les problèmes traités soient décrits par des systèmes d'équations. Il s'agit de méthodes statistiques qui calculent un résultat et une incertitude statistique associée.

En science des matériaux [Lem04], le code SRIM [Zie] permet de simuler des dommages d'irradiation par la méthode de Monte-Carlo. Ces dommages sont la conséquence d'interactions d'ions incidents avec la matière, qui réalisent des cascades de déplacements d'atomes dans des matériaux. Ce code calcule des interactions électroniques et nucléaires, ainsi que les pertes d'énergie d'ions de masses et d'énergies données au sein d'un matériau.

En neutronique les codes de calcul Monte-Carlo tels que TRIPOLI4 [BMN94] et MCNP [Bri93] calculent des taux de réactions[3] : c'est-à-dire un nombre d'événements d'un certain type (fission, absorption, diffusion...) qui se déroulent dans un volume précis pendant un certain temps. Ces méthodes de Monte-Carlo permettent de simuler des configurations complexes, en trois dimensions, sans pour autant nécessiter d'une discrétisation du domaine d'étude en espace, en angle ou en énergie des particules.

La méthode de Monte-Carlo consiste à approcher des espérances mathématiques de variables aléatoires en simulant un grand nombre d'expériences issues de tirages aléatoires. La précision de calcul dépend alors du nombre d'expériences simulées. Ces expériences sont, pour le cas de la neutronique, les histoires des particules dans le réacteur, de leur *naissance* (source externe, neutron de fission, ...) à leur *disparition* (capture par un noyau ou fuite du réacteur).

[3] Les taux de réactions sont des grandeurs physiques observables et définies pas $R_i = \Sigma_i \Phi$; où Σ_i est la section efficace du milieu renseignant sur la probabilité d'une interaction des neutrons avec le milieu , et Φ désigne le flux de neutrons dans le volume considéré.

La première étape d'un calcul Monte-carlo consiste donc à tirer aléatoirement un état, pour chaque particule suivie, caractérisé par une position dans l'espace $\overrightarrow{r}$, une énergie E[4] et un angle $\overrightarrow{\Omega}$. Ce tirage aléatoire dépend alors de la répartition en espace, en énergie et en angle, définie en amont du calcul.
A chaque milieu du domaine est associé un ensemble de caractéristiques neutroniques appellées sections efficaces. Les sections efficaces Σ_i sont fonction de l'énergie du neutron incident, de l'angle et de la température (fixée) du milieu. Dans le processus d'un calcul Monte-Carlo, connaissant l'état ($\overrightarrow{r}$, E, $\overrightarrow{\Omega}$) de chaque neutron et la répartition des sections efficaces pour chaque milieu, une deuxième étape consiste à tirer aléatoirement une trajectoire associée à un neutron. Cette trajectoire transporte le neutron dans le cœur au lieu d'une prochaine interaction.
Les neutrons ainsi transportés, il s'agit alors, dans une troisième étape, de tirer aléatoirement le type d'interaction qu'ils subissent. Dans le cas d'un milieu non fissile, on définit la section efficace totale Σ_t, comme étant la somme des sections d'absorption Σ_a et de diffusion Σ_s. Cette fois-ci, les sections d'absorption Σ_a et de diffusion Σ_s sont distinguées de la section efficace totale Σ_t, telles que $\Sigma_t = \Sigma_a + \Sigma_s$. Le tirage aléatoire consiste à choisir si le neutron subit une absorption ou une diffusion. Pour ce faire, un tirage aléatoire dans une loi uniforme d'un nombre s est réalisé, et si $s \leq \frac{\Sigma_a}{\Sigma_t}$ alors on considère que l'interaction est une absorption, sinon on considère qu'il s'agit d'une diffusion, impactant la direction $\overrightarrow{\Omega}$ et l'énergie E du neutron. Les grandeurs physiques calculées (appellées *scores*) sont enregistrées au fur et à mesure des simulations de particules.

La mise en données d'un calcul Monte-Carlo nécessite peu de simplifications des problèmes de physique. Ces schémas de calcul sont souvent utilisés comme schémas de référence par rapport à des simulations nécessitant des approximations[5]. Les jeux de données d'une modélisation Monte-Carlo consistent en général à décrire la géométrie (qui peut être une géométrie CAO), à associer de matériaux à des volumes, à initialiser les sources[6], à spécifier le traitement des résultats, et dans certains cas à définir un biaisage pour amener artificiellement des particules en des lieux d'intérêts du domaine d'étude.

Ainsi, la construction des géométries de calcul, lors de la mise en données, n'est pas forcément un maillage. On voit généralement apparaître des modèles géométriques proches des représentations de géométries de type CSG [Hof89] (voir paragraphe 3.2). Par ailleurs, il arrive que la géométrie soit sur-découpée pour obtenir des résultats sur des sous-volumes de la géométrie.

Différences fondamentales des méthodes déterministes et de Monte-Carlo

A la différence des codes de calcul déterministes, les codes Monte-Carlo ne supposent pas d'approximation. Ils permettent de valider numériquement des hypothèses et des simplifications introduites par les calculs déterministes. Du point de vue de l'utilisateur, une grande différence entre les deux méthodes est le temps de calcul : les calculs déterministes sont en général plus rapides que les calculs Monte-Carlo, dont le principe

[4] L'énergie cinétique du neutron est équivalente à sa vitesse selon la relation $E = \frac{1}{2}.m_n.V^2$

[5] Réalisées par exemple avec des méthodes de calcul déterministes.

[6] Pour la neutronique, cela consiste à fournir au code les premières particules

est de simuler un grand nombre d'expériences. Les physiciens font appel aux calculs parallèles[7] pour réduire les temps de calcul.
Les codes Monte-Carlo permettent de traiter des géométries 3D très complexes, les difficultés provenant alors de la description de la géométrie pertinente pour les effets à étudier.

Les modèles géométriques utilisés par les codes de type Monte-Carlo se rapprochent des modèles CSG ; les géométries utilisées par les codes déterministes (éléments finis, différences finies, volumes finis et caractéristiques) sont discrétisées et représentées par un maillage qui se rapproche des modèles B-Rep.

S'il existe un modèle générique permettant la modélisation d'un dispositif d'irradiation dans chaque discipline, ce modèle étant indépendant de la méthode de calcul et des codes de calcul, alors du point de vue géométrique, il s'agit d'un modèle hybride CSG/B-Rep (voir paragraphe 3.3).

2.1.2 Maillages des méthodes de calcul déterministes

Lors de l'utilisation de méthodes de calcul déterministes, il est nécessaire de réaliser une discrétisation du domaine d'étude, appelée maillage. Dans le cas de domaines géométriques, ces maillages sont calculés et générés par des logiciels appelés mailleurs. Les numériciens paramétrisent les mailleurs en fonction du problème de modélisation visé et de l'objet géométrique à discrétiser, avec différents algorithmes de maillage et différentes hypothèses (ou contraintes). Une définition générale d'un maillage, tirée de [FG99], serait la suivante :

τ_h est un maillage du domaine Ω si

- $\Omega = \overline{\bigcup_{K \in \tau_h}^{\circ} K}$;
- L'intérieur de tout élément K de τ_h est non vide ;
- L'intersection de l'intérieur de deux éléments est vide (ce qui interdit les chevauchements d'éléments).

où K est un simplexe, $\overline{u}$ est l'adhérent du u et $\overset{\circ}{u}$ est l'intérieur de u.

Rappelons brièvement la définition d'un simplexe : on considère $d+1$ points a_j non situés dans un même hyperplan. On appelle d-simplexe K de sommets a_j l'enveloppe convexe des points a_j. En dimension 2 un simplexe est un triangle, en dimension 3 un simplexe est un tétraèdre.
En topologie, l'adhérence d'une partie X d'un espace topologique E est le plus petit ensemble fermé de E qui contient X ; L'intérieur d'un ensemble topologique X est l'ensemble de tous les points p (intérieurs de X) tels qu'il existe un voisinage de p inclus dans X.

Pour les méthodes différences finies, les maillages sont dit structurés ou en grilles. La

[7] De l'ordre de 60 processeurs pour simuler le RJH en 3D pour des résultats sur tous les milieux fissiles.

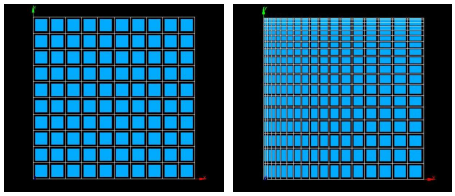


FIG. 2.1 – Exemples de maillages structurés, régulier à gauche et irrégulier à droite.

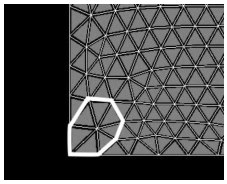
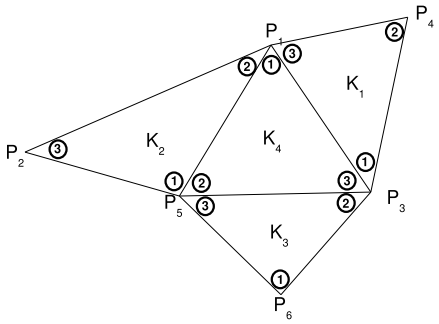


FIG. 2.2 – Exemple d'un maillage non structuré.

connectivité[8] est dite de type différence finie. Les maillages structurés peuvent être réguliers, dans ce cas la discrétisation est uniforme, ou irréguliers (voir figure 2.1). Dans la pratique, ces maillages se caractérisent par le fait que chacun des éléments peut être référencé par un unique n-uplet d'indices, avec n la dimension d'espace du maillage. Par exemple, pour un maillage de dimension 2, un couple d'indices suffit pour référencer un élément du maillage.

Les méthodes éléments finis permettent l'utilisation de maillages dit non structurés. Il s'agit de maillages dont la connectivité est quelconque. Par exemple, le maillage de la figure 2.2 est non structuré. Il est formé de triangles dont les sommets peuvent être communs à plus de 6 éléments (éléments entourés). Ces maillages présentent l'avantage d'être plus souples à mettre en œuvre, vis-à-vis des contraintes et algorithmes des mailleurs appliqués sur des géométries complexes. Cependant, ces maillages ne permettent pas de référencer un élément par un n-uplet, commme les maillages structurés le permettent. Dans la pratique, des méthodes de recherche par voisinage basées sur les propriétés de connectivité des éléments sont utilisées pour référencer un élément d'un maillage non structuré.



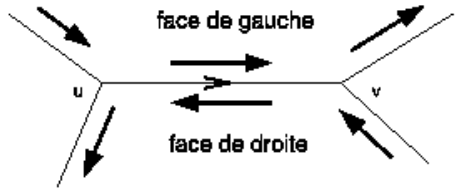
K	*P*
4 0 0	1 3 4
4 0 0	1 2 5
4 0 0	3 5 6
2 3 1	1 5 3

graphe d'adjacence

FIG. 2.3 – Maillage composé de quatre triangles et son graphe d'adjacence. Les indices locaux de chaque triangle sont entourés.

L'information de connectivité des éléments peut être décrite par différentes structures

[8] La connectivité d'un maillage définit le type de connexions entre ses sommets.



arête	sommets de	sommets à	faces droite	faces gauche	sens direct préc.	sens direct suiv.	sens indirect préc.	sens indirect suiv.
e	6	7	E	B	q	t	r	u
u	7	11	E	F	f	i	e	h
h	10	11	H	E	v	s	u	r
r	10	6	E	D	g	d	h	c

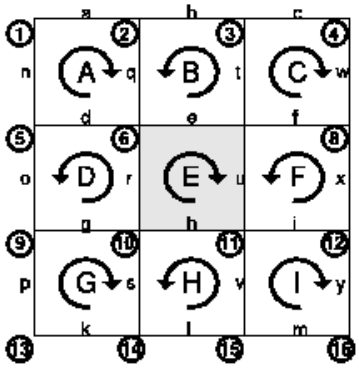


FIG. 2.4 – Représentation et exemple d'un maillage par arêtes étoilées (winged-edge).

topologiques. Une première représentation possible de cette information serait un graphe d'adjacence. Dans le cas d'un maillage de triangles, en 2D, cela consiste à renseigner pour chaque triangle le couple de triplets de sommets-voisins. Par exemple, pour l'élément K_4 de la figure 2.3, le triplet de sommets est (P_1, P_5, P_3) et le triplet de voisins est (K_2, K_3, K_1), pour définir l'extérieur on pourra utiliser l'indice d'élément 0. Cette représentation présente l'avantage d'être concise, mais n'est pas suffisamment riche pour accéder aisément aux voisinages d'un élément.

Une représentation plus riche se base sur des triplets de sommets et des matrices de connexions. A chaque triangle K_n est associé une matrice 3×3, et un indice local est attribué à chacun de ses sommets. La matrice C^{K_n} est alors construite de la manière suivante :

- on indice localement à i l'arête opposée au sommet dont l'indice local est i ;
- un coefficient diagonal c_{ii} donne l'indice local du sommet opposé du triangle voisin de K_n par l'arête i ;
- un coefficient c_{ij} donne l'indice, dans le triangle voisin de K_n par l'arête i, du sommet j de K_n.

Ainsi, le triangle K_4 de la figure 2.3 est associé à la matrice $C^{K_4} = \begin{bmatrix} 1 & 3 & 2 \\ 3 & 2 & 1 \\ 2 & 1 & 3 \end{bmatrix}$

Cette représentation, plus riche que la première, permet d'accéder simplement aux informations de voisinage entre éléments. Toutefois, les matrices C^{K_n} n'ayant pas de propriétés particulières permettant d'être réduites, cette représentation se trouve plus gourmande en mémoire.

Une autre solution pour représenter un maillage est de le regarder sous l'angle de ses arêtes. On trouve par exemple des méthodes basées sur une structure de type *winged-edge* (ou arête étoilée) [Hof89]. Ces méthodes permettent essentiellement de tourner autour des arêtes d'une face, trouver deux faces adjacentes via une arête, trouver toutes les arêtes incidentes en un sommet. Il s'agit d'une struture de données d'une grande richesse.

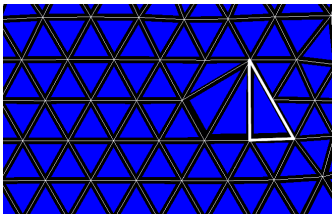


FIG. 2.5 – Exemple d'un maillage non conforme

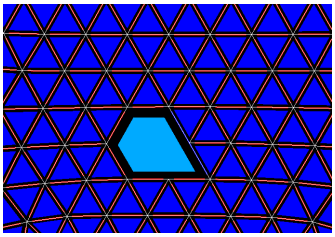


FIG. 2.6 – Exemple d'un maillage mixte non conforme

La figure 2.4, présente le principe de représentation par arêtes étoilées ainsi qu'un exemple sur un maillage régulier, limité à la description des arêtes de l'élément E. Dans cette représentation chaque arête est orientée, tel un vecteur, et un sens de rotation est associé à chaque face de manière à ce que deux faces adjacentes soient parcourues dans des sens différents.

L'avènement de la programmation orientée objet, simplifie la mise en œuvre d'une autre représentation dite hiérarchique [FG99]. De manière schématique, cette représentation renseigne sur la hiérarchie entre les entités de ces maillages, dans l'ordre de leurs dimensions. On a alors les liens directs de type : sommets $\rightarrow$ arêtes $\rightarrow$ faces $\rightarrow$ éléments et les liens inverses éléments $\rightarrow$ faces $\rightarrow$ arêtes $\rightarrow$ sommets.

Cette représentation, plus générale, est applicable tant aux triangulations (simpliciales) qu'aux maillages arbitraires. Elle permet de décrire des maillages dits *non conformes*, comme ceux présentés par les figures 2.5 et 2.6. Par définition ([FG99]), un maillage est conforme si l'intersection de deux éléments est soit l'ensemble vide, soit un sommet, un bord ou une face (selon que l'on travaille en dimension 2 ou 3) commun aux deux éléments. Toutefois la plupart des schémas numériques utilisant un maillage comme support spatial supposent que ce maillage est conforme.

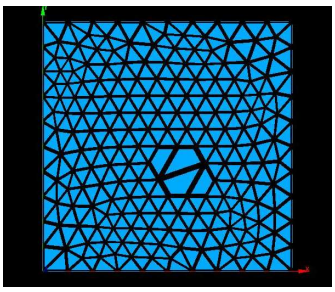


FIG. 2.7 – Exemple d'un maillage mixte triangle/quadrangle.

Enfin, lorsque les éléments d'un maillage sont de natures différentes, on parle alors de maillage mixte (voir figures 2.6 et 2.7).

Les modélisations physiques réalisées dans le cadre de la thèse sont effectuées sur des maillages géométriques de dimension 1,2 ou 3. Les bords de mailles sont en général des segments. Cependant la méthode des caractéristiques du code Apollo2 accepte des maillages mixtes et les bords des mailles peuvent être des segments, des arcs de cercles ou des développantes de cercles (figure 2.8).

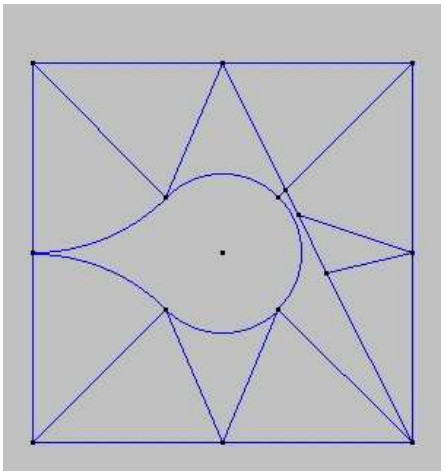


FIG. 2.8 – Exemple d'un maillage mixte contenant des bords de mailles en arc de cercle, utilisé par la méthode des caractéristiques en neutronique.

Pour les méthodes de calcul déterministes, les méthodes éléments finis et différences finies sont les plus couramment utilisées. Dans nos travaux, seules les simulations neutroniques des cœurs de réacteurs expérimentaux utilisent la méthode des caractéristiques. Cette méthode est utilisée pour résoudre l'équation de transport des neutrons dont les principes sont décrits ci-après.

2.2 Principes de neutronique

2.2.1 Equation de BOLTZMANN

Les simulations de neutronique réalisées dans les travaux de thèse, permettent d'étudier le comportement de la population de neutrons ([Reu03], [Sal02]) dans un réacteur expérimental et les conditions d'irradiations neutroniques des dispositifs expérimentaux. Pour représenter cette population de neutrons, sept variables sont nécessaires :

- trois variables d'espace (dans le cas de calculs à trois dimensions), pour repérer la position des neutrons ;
- trois variables de vitesse qui caractérisent l'état des neutrons (une variable v pour le module de la vitesse et deux autres pour l'angle solide $\overrightarrow{\Omega}$) ;
- une variable temporelle.

Les neutroniciens caractérisent la population de neutrons par une fonction-densité : le *flux* neutronique $\Phi = nv$, avec n la densité de neutrons par unité de volume, par unité de vitesse v par unité d'angle $\overrightarrow{\Omega}$ et par l'instant t considéré.

La densité de neutrons dans un réacteur étant élevée (de l'ordre de $10^8 n.cm^{-3}$), en négligeant les interactions neutron-neutron[9], la population de neutrons peut être considérée comme un gaz parfait. L'équation régissant le transport des neutrons dans le cœur du réacteur est alors l'équation de BOLTZMANN.

Cette équation est établie par un bilan du nombre de neutrons dans un volume élémentaire de l'espace :

$$\begin{aligned}\frac{1}{v}\frac{\partial}{\partial t}\Phi\left(\overrightarrow{r},v,\overrightarrow{\Omega},t\right) = & \overbrace{-div\left[\overrightarrow{\Omega}\cdot\Phi\left(\overrightarrow{r},v,\overrightarrow{\Omega},t\right)\right]}^{Deplacements} \\ & \overbrace{-\Sigma_t\left(\overrightarrow{r},v,\overrightarrow{\Omega},t\right)\Phi\left(\overrightarrow{r},v,\overrightarrow{\Omega},t\right)}^{Chocs} \\ & \overbrace{+\int_{4\pi}d\overrightarrow{\Omega}'\int_0^{\infty}dv'\left[\Sigma_s\left(\overrightarrow{r},\left(v',\overrightarrow{\Omega}'\right)\rightarrow\left(v,\overrightarrow{\Omega}\right),t\right)\Phi\left(\overrightarrow{r},v',\overrightarrow{\Omega}',t\right)\right]}^{Transferts} \\ & \overbrace{+S\left(\overrightarrow{r},v,\overrightarrow{\Omega},t\right)}^{Sources}\end{aligned}$$

Le terme du premier membre correspond à la variation du nombre de neutrons contenus dans un élément de volume, et les termes du second membre correspondent aux disparitions et aux arrivées de neutrons :

- le terme de *déplacements* est un bilan des neutrons qui entrent et qui sortent du volume ;
- le terme de *chocs* correspond à l'ensemble des disparitions par choc avec un noyau du volume (changement de direction et de vitesse suite à une collision avec un noyau) ;
- le terme de *transferts* représente les arrivées de neutrons de vitesse v' et d'angle $\overrightarrow{\Omega}'$ dans l'intervalle de vitesse v et d'angle $\overrightarrow{\Omega}$, par ralentissement, ou par *upscattering* (gain d'énergie), au cours d'un choc.
- le terme *sources* est l'ensemble des neutrons produits dans le volume (par exemple par fission).

2.2.2 Principes de résolution

L'impossibilité pratique d'une résolution analytique de l'équation de Boltzmann, impose aux physiciens de simplifier et de discrétiser les différents domaines d'études :

- spatial, en résolvant le problème sur un maillage de la géométrie ;

[9] On trouve 10^{22} noyaux atomiques par cm^3 dans la matière, de ce fait un neutron à 10^{14} fois plus de chance de rencontrer un noyau qu'un neutron.

- angulaire, en ne considérant qu'une quantité discrète de directions ;
- énergétique, en découpant le domaine énergétique en groupes d'énergies judicieusement choisis (ce découpage en groupes est sujet à de nombreux travaux de recherche).

La discrétisation en groupes énergétiques est traduite par une tabulation $\Sigma_{i,g}$ des sections efficaces $\Sigma_i(E)$ (où E est l'énergie cinétique d'un neutron de vitesse v et l'indice i correspond au type d'interaction), de telle sorte que les taux de réactions soient conservés :

$$\Phi_g\left(\overrightarrow{r},\overrightarrow{\Omega},t\right) = \int_{E_g}^{E_{g+1}} \Phi\left(\overrightarrow{r},E,\overrightarrow{\Omega},t\right) dE \tag{2.1}$$

$$\Sigma_{i,g}\left(\overrightarrow{r},\overrightarrow{\Omega},t\right) = \frac{\int_{E_g}^{E_{g+1}} \Sigma_i\left(\overrightarrow{r},E,\overrightarrow{\Omega},t\right)\Phi\left(\overrightarrow{r},E,\overrightarrow{\Omega},t\right) dE}{\Phi_g\left(\overrightarrow{r},\overrightarrow{\Omega},t\right)} \tag{2.2}$$

Selon l'échelle de temps considérée, il est possible de séparer les problèmes temporels en différentes catégories :

- les problèmes stationnaires : en fonctionnement normal, un réacteur peut être modélisé par un régime stationnaire, c'est-à-dire indépendant du temps. Ainsi, le premier terme de l'équation de Boltzmann s'annule et seules 6 variables au lieu de 7 sont à traiter ;
- les problèmes d'évolution du combustible : la fission des noyaux des milieux combustibles implique une modification de la composition isotopique, qui influe sur les sections efficaces du milieu. Ce problème est donc dépendant du temps d'irradiation que les physiciens ramènent à un taux de combustion, appelé *burn-up* et exprimé en Mégawatts par jour et par tonne d'isotopes fissiles.

Dans la pratique, ces deux problèmes sont traités séquentiellement, en effectuant dans un premier temps un calcul stationnaire qui permet de calculer les flux neutroniques par groupe d'énergie. Il s'ensuit un calcul d'évolution des milieux combustibles, tenant compte des taux de réactions (réactions de fission) déduits du précédent calcul de flux Φ. On parle alors d'approximation quasi-stationnaire.

On constate alors que le traitement de la neutronique se ramène à réaliser un couplage entre le calcul de flux et le calcul d'évolution. Les schémas de calcul sur lesquels les travaux de thèse se sont appuyés, traitant de la neutronique des cœurs RJH (formulaire HORUS3D [HAB+05], [WAB+04]) et OSIRIS (formulaire ANUBIS [SBd+07], [BSA+06]), sont architecturés sur un schéma de couplage dit explicite faible (voir figure 2.12 présentée au paragraphe 2.4). Le principe de ces schémas sera présenté lors de l'utilisation dans le chapitre 9.

2.2.3 Influence des autres disciplines

D'autres disciplines peuvent influencer les grandeurs de neutronique, soit en modifiant les caractéristiques neutroniques des milieux du domaine, c'est-à-dire les sections efficaces Σ_i, soit en modifiant la géométrie du domaine. Les résolutions déterministes de l'équation de Boltzmann sont basées sur un maillage du domaine géométrique, et les modifications de la géométrie ont une influence sur ce maillage. Les modifications des

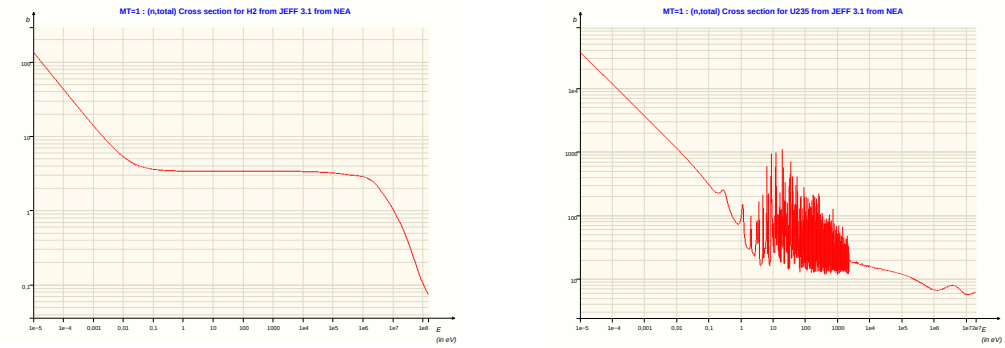


FIG. 2.9 – Section efficace microscopique totale du ^2H à gauche, et du ^{235}U (résonnante) à droite.

sections efficaces des milieux, présentées en annexe B, peuvent quant-à-elles provenir de plusieurs paramètres dont en particulier la variation des compositions isotopiques et des températures des milieux.

Dans le cas des méthodes déterministes, nous avons vu que la résolution de l'équation de Boltzmann est réalisée après la mise en groupes énergétiques des sections efficaces (voir les équations 2.1 et 2.2). Cette étape de mise en groupes nécessite de connaître a priori le flux Φ. Cependant, ce flux est la grandeur que l'on cherche à obtenir par la résolution de l'équation de Boltzmann.

En général, une approximation de ce flux de pondération est réalisée, en considérant un flux donné, proche du réacteur à calculer. Il reste toutefois une difficulté à lever dans le cas des résonances qui nécessite un traitement numérique particulier appelé autoprotection.

En effet, les sections efficaces de certains isotopes lourds (isotope de nombre de masse élevé), tels que le ^{235}U, présentent des résonances (voir figure 2.9). Au niveau de ces résonances, une augmentation brutale de la section efficace induit une diminution du flux dans le groupe énergétique. La mise en groupe nécessite alors un calcul exact du flux de pondération qui dépend de la géométrie et des matériaux. Toutes ces étapes sont mises en œuvre dans un formalisme d'autoprotection spécifique au réacteur étudié.

Nous avons vu que les sections efficaces microscopiques dépendent de l'énergie des neutrons incidents, c'est-à-dire de leur vitesse. Il a été considéré jusqu'à présent qu'un noyau percuté par un neutron était initialement immobile. Cependant, à l'échelle des neutrons et des noyaux, la température d'un milieu se caractérise par une agitation thermique des noyaux. Si l'on tient compte, lors d'une interaction neutron-noyau, de la faible vitesse du noyau cible au moment de l'impact, alors la vitesse relative du neutron par rapport au noyau est légèrement modifiée. La section efficace, fonction de la vitesse (relative) est donc modifiée : c'est l'effet Doppler.

L'effet Doppler se caractérise par une dilatation en énergie des sections efficaces, et donc des largeurs des résonances des sections. En pratique on constate une augmentation

de l'absorption résonnante des noyaux lors d'une augmentation de la température (et donc de la fréquence d'agitation thermique des noyaux.). Cette augmentation de la température ainsi que le comportement du fluide caloporteur qui s'écoule le long des crayons sont également pris en compte par la thermohydraulique dont nous allons voir les principales notions.

2.3 Principes de thermohydraulique

L'objectif de la thermohydraulique dans l'étude des réacteurs nucléaires est double. D'une part il s'agit d'évaluer le comportement des écoulements de fluides (que ce soit de l'eau pour les REP ou du sodium pour les RNR), c'est-à-dire la distribution des vitesses dans un domaine ; d'autre part il s'agit d'évaluer la distribution de température et de pression.

Dans la majeure partie des cas, notamment ceux traités dans cette étude, les écoulements étudiés sont considérés monophasiques tout au long de l'expérience. Il est cependant envisageable de traiter des écoulements diphasiques liquide-vapeur.

2.3.1 Equation de Navier-Stokes

La discipline de la physique qui traite de tels écoulements est la mécanique des fluides. Les équations régissant les écoulements de fluides visqueux (que ce soit des gaz ou des liquides) sont les équations de Navier-Stokes.

La formulation différentielle de ces équations est la suivante :

– Équation de continuité (ou équation de bilan de la masse)

$$\frac{\partial \rho}{\partial t} + \overrightarrow{\nabla} \cdot (\rho \vec{v}) = 0 \tag{2.3}$$

– Équation de bilan de la quantité de mouvement

$$\frac{\partial (\rho \vec{v})}{\partial t} + \overrightarrow{\nabla} \cdot (\rho \vec{v} \otimes \vec{v}) = -\overrightarrow{\nabla p} + \overrightarrow{\nabla} \cdot \overrightarrow{\overrightarrow{\tau}} + \rho \vec{f} \tag{2.4}$$

– Équation de bilan de l'énergie

$$\frac{\partial (\rho e)}{\partial t} + \overrightarrow{\nabla} \cdot [\, (\rho e + p)\, \vec{v}\,] = \overrightarrow{\nabla} \cdot \left(\overrightarrow{\overrightarrow{\tau}} \cdot \vec{v} \right) + \rho \vec{f} \cdot \vec{v} - \overrightarrow{\nabla} \cdot \vec{q} + r \tag{2.5}$$

Avec :

– t représente le temps ;
– ρ désigne la masse volumique du fluide ;
– $\vec{v}$ désigne la vitesse ;
– p désigne la pression ;
– $\overrightarrow{\overrightarrow{\tau}} = (\tau_{i,j})_{i,j}$ est le tenseur des contraintes visqueuses ;

- $\vec{f}$ désigne la résultante des forces massiques s'exerçant dans le fluide, souvent simplifiée à la force de pesanteur $\vec{g}$;
- e est l'énergie totale par unité de masse ;
- $\vec{q}$ est le flux de chaleur perdu par conduction thermique, et la loi de Fourier donne $\vec{q} = -\lambda\overrightarrow{\nabla}T$ avec λ la conductivité thermique et T la température ;
- r représente la perte de chaleur volumique due au rayonnement.

2.3.2 Principes de résolution

En général, les fluides sont considérés incompressibles (au contraire des gaz qui sont compressibles) d'où $\frac{\partial\rho}{\partial t} = 0$. De plus ρ est indépendant des variables d'espace ainsi $\overrightarrow{\nabla} \cdot (\rho\vec{v}) = \rho\overrightarrow{\nabla} \cdot \vec{v}$.
L'équation 2.3 devient alors :

$$\overrightarrow{\nabla} \cdot \vec{v} = 0 \tag{2.6}$$

et l'équation 2.4 devient :

$$\frac{\partial\vec{v}}{\partial t} + \left(\vec{v} \cdot \overrightarrow{\nabla}\right)\vec{v} = -\frac{1}{\rho}\overrightarrow{\nabla p} + \nu\nabla^2\vec{v} + \vec{f} \tag{2.7}$$

avec $\nu = \frac{\mu}{\rho}$ appelée viscosité cinématique et μ la viscosité dynamique du fluide.

La résolution de l'équation de Navier-Stokes n'est pas triviale : à la complexité inhérente aux équations aux dérivées partielles s'ajoute celle de la non-linéarité introduite par le terme d'advection de l'accélération $\left(\vec{v} \cdot \overrightarrow{\nabla}\right) \cdot \vec{v}$.

Selon le régime d'écoulement étudié (laminaire, transitoire ou turbulent), différentes simplifications de l'équation 2.7 sont réalisables. En mécanique des fluides, le régime d'écoulement est caractérisé par le nombre de Reynolds Re tel que :

$$Re = \frac{V \cdot L}{\nu}$$

avec V la vitesse du fluide et L une distance caractéristique du phénomène (par exemple le diamètre d'une conduite).

Ainsi pour des valeurs faibles du nombre de Reynolds (inférieures à 1), le régime est laminaire et l'écoulement dit de Stokes. Pour des valeurs élevées du nombre de Reynolds (en général supérieures à 2000), le régime est dit turbulent, le fluide peut être considéré comme parfait (pas d'effet de viscosité et de conductivité thermique). Entre ces deux régimes, l'écoulement est transitoire.

Ces différents régimes d'écoulement sont caractérisables par leurs profils de vitesses, présentés par la figure 2.10. Intuitivement, en écoulement laminaire, les parois (ainsi que tout autre objet introduit dans l'écoulement) ont une influence sur l'écoulement plus importante qu'en régime turbulent.

Ces deux régimes permettent de simplifier l'équation 2.7 différemment. Ainsi le terme d'advection peut-être négligé en écoulements de Stokes (en considéant $Re \ll 1$) et

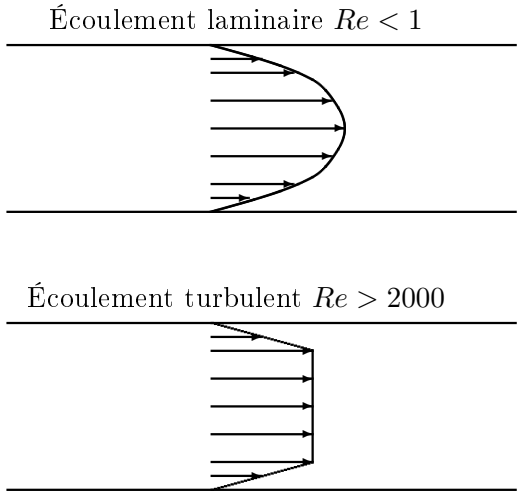


FIG. 2.10 – Profils de vitesse d'un fluide incompressible dans une conduite selon le régime d'écoulement.

l'équation 2.7 devient :

$$\frac{\partial \vec{v}}{\partial t} = -\frac{1}{\rho}\overrightarrow{\nabla p} + \nu \nabla^2 \vec{v} + \vec{f}$$

Dans le cas d'une hypothèse de fluide parfait les effets de viscosité sont négligés, l'équation 2.7 devient dans ce cas :

$$\frac{\partial \vec{v}}{\partial t} + \left(\vec{v} \cdot \overrightarrow{\nabla}\right) \vec{v} = -\frac{1}{\rho}\overrightarrow{\nabla p} + \vec{f}$$

Les résolutions numériques sont le plus souvent réalisées, après linéarisation des équations, par des méthodes différences finies ou éléments finis, voire volumes finis, dont les maillages sont décrits au paragraphe 2.1.2.

2.3.3 Influence des autres disciplines

Comme pour la neutronique, la déformation de la géométrie influe sur le calcul thermohydraulique. Ces déformations peuvent en partie être induites par des gradients de température présents dans les milieux solides, régis par la thermique. Cette discipline permet d'étudier la distribution de températures dans des milieux, à partir de l'équation de la chaleur 2.8.

$$\rho c \frac{\partial T}{\partial t} = \overrightarrow{\nabla} \cdot \left(\lambda \overrightarrow{\nabla T}\right) \tag{2.8}$$

avec c la capacité thermique massique.

Cette dernière équation peut être traitée comme un problème de diffusion en réalisant l'approximation suivante :

$$\frac{\partial T}{\partial t} = D \Delta T$$

avec $D = \frac{\lambda}{\rho c}$ le coefficient de diffusion thermique.

En pratique, les codes de calcul de thermohydraulique, proposent des méthodes de couplage implicite (voir paragraphe 2.4) des équations de Navier-Stokes avec l'équation de la chaleur. L'influence des autres disciplines provient généralement de modifications des conditions aux limites des calculs, par exemple la modification d'un champ de température forcé sur une paroi. On s'aperçoit qu'il existe diverses manières de coupler des problèmes de physique, le paragraphe suivant en présente quelques notions.

2.4 Notions générales sur le couplage

Le problème exact de l'étude d'un dispositif d'irradiation consisterait à calculer les paramètres de la neutronique, de la thermohydraulique et de la thermique dans un même système dont les équations seraient dépendantes les unes des autres. Ce type de couplage dit *implicite* nécessite de coupler les disciplines au niveau des équations de la physique. La complexité de mise en pratique d'un tel problème impose de se pencher sur une résolution de chacune des disciplines de manière indépendante. Dans cette optique, les schémas de calcul ne résolvent qu'un aspect des phénomènes physiques, et on parle de couplage *explicite* : il s'agit d'une résolution séquentielle et automatique des opérations, où chaque schéma modifie les données d'entrée du code suivant. Il peut y avoir plusieurs itérations jusqu'à convergence des résultats. Ce couplage possède l'avantage d'utiliser les codes déjà existants. Traditionnellement, on différencie deux types de couplages explicites : les couplages explicites forts et les couplages explicites faibles.

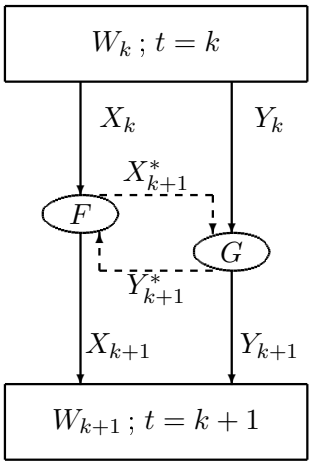


FIG. 2.11 – Schéma de couplage explicite fort.

Considérons F et G deux schémas de calcul que l'on souhaite coupler, k un indice de pas de temps, X_k les données d'entrée nécessaires au schéma F pour calculer les données X_{k+1}, de même que Y_k les données d'entrée nécessaires au schéma G pour calculer les données Y_{k+1} et $W_k = X_k \cup Y_k$.

Dans le cas d'un couplage explicite fort, présenté par la figure 2.11, F et G échangent mutuellement des informations, éventuellement plusieurs fois lors d'un pas de temps.

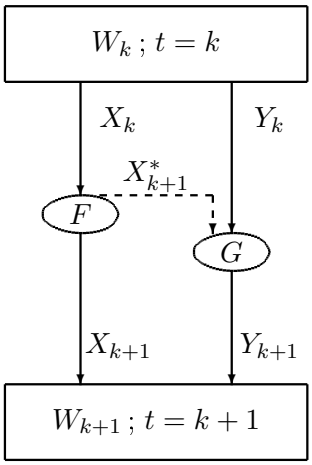


FIG. 2.12 – Schéma de couplage explicite faible.

Ces échanges d'informations conduisent à des itérations jusqu'à la convergence[10] des informations échangées. Après convergence, les paramètres sont modifiés et le calcul itère pour le pas de temps suivant.

Lors d'un couplage explicite faible, présenté par la figure 2.12, le premier schéma F modifie les paramètres d'entrée de son successeur G. Les paramètres calculés par G seront introduits dans le calcul pour le pas de temps suivant. Ce type de couplage peut être utilisé s'il est considéré que Y_{k+1} n'a pas (ou peu) d'impact sur X_{k+1}. On parle dans ce cas d'enchaînement (ou chaînage) de calculs.

Les développements réalisés durant la thèse ont pour objectif de fournir des outils adaptés au couplage explicite des schémas de calcul et de permettre a minima un enchaînement automatisé de simulations numériques de chaque discipline. En considérant que les schémas de calcul sont développés, la difficulté de tels couplages réside dans la maîtrise des données échangées. Un cas d'application, présenté en annexe A, traitant d'un exemple de couplage réalisé analytiquement, permet de référencer le type de données qui peuvent être échangées dans ce cas précis. La géométrie des objets fait partie de ces données communes aux différents modèles de physique. L'aspect pluridisciplinaire de cette thèse réside, justement, dans le fait de proposer un modèle de données alliant des caractéristiques physiques associées à une représentation géométrique, dont les différents concepts sont détaillés dans le chapitre 3.

[10] En s'assurant de la convergence du système.

Chapitre 3

Modèles géométriques

Comme le présente le chapitre 2, les simulations numériques sont effectuées sur des domaines géométriques, ou des discrétisations de ces domaines appelées maillages. Ces domaines géométriques sont décrits par des modèles numériques, dont l'objectif est de se rapprocher au mieux de la géométrie réelle. En pratique, le modèle géométrique numérique est une simplification satisfaisante de la géométrie réelle, vue selon une représentation spécifique. L'objectif de ce chapitre est de présenter les moyens de représentation des géométries utilisées dans les outils de CAO. Deux approches ont connu une très large diffusion : le modèle B-Rep (Boundary Representation) qui représente un objet par les bords séparant son intérieur de son extérieur, et le modèle CSG (Constructive Solid Geometry) qui représente un historique de construction définissant l'intérieur d'un objet.

3.1 Modèle B-Rep

3.1.1 Principes élémentaires

Dans un modèle B-Rep (*Boundary Representation* ou *modèle par les frontières*), le système *connaît* la peau de l'objet et le côté où se trouve la matière. Intuitivement, la conception d'un modèle B-Rep consiste à *coudre* la géométrie avec des entités géométriques de dimensions différentes. Pour un solide de dimension 3, les entités géométriques sont :

- des noeuds, entités de dimension 0, qui fixent une position dans l'espace ;
- des courbes, entités de dimension 1, qui permettent de décrire les bords d'une face ;
- des surfaces, entités de dimension 2, qui permettent d'approcher la peau du solide ;

Deux familles d'informations sont conservées dans un modèle B-Rep :

- l'information géométrique : elle permet de spécifier la position des objets dans l'espace, et de décrire les formes géométriques.
- l'information topologique : elle relie les différentes entités entre elles.

Par exemple pour représenter la face présentée par la figure 3.1, les noeuds n_o à n_4 positionnent la face dans l'espace en conservant les coordonnées. Les courbes a_0 à a_4

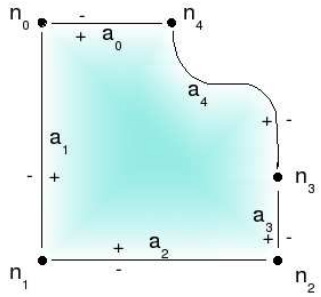


FIG. 3.1 – Représentation d'une face par un modèle B-Rep.

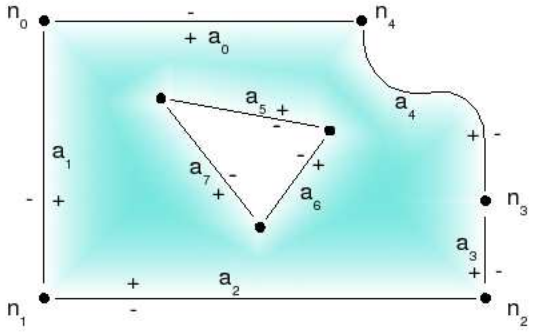


FIG. 3.2 – Représentation B-Rep d'une face trouée.

relient chacune deux noeuds, c'est l'information topologique. De plus, ces courbes sont caractérisques d'une forme géométrique : des segments pour a_0, a_1, a_2, a_3 et une courbe qui peut être définie par des polynômes (par exemple des courbes de type Bézier ou B-Spline [Dan07]) pour a_4 ; c'est l'information géométrique. La face est caractérisée par un plongement géométrique (c'est-à-dire une forme) et des limites définies par un bord fermé composé des courbes a_0 à a_4 : c'est l'information géométrique. La face relie donc les courbes connexes a_0 à a_4, orientées de manière à définir l'intérieur (côté +) et l'extérieur (côté −) de la face ; c'est l'information topologique. Dans le cas de modélisation d'une face *trouée*, la face est caractérisée par plusieurs bords fermés (figure 3.2), et généralement, le sens de parcours des entités de bords n'est pas anodin : les entités de bords appartenant au périmètre extérieur sont parcourues dans le sens direct (a_0 à a_4), celles appartenant à des périmètres intérieurs (des trous) sont parcourues dans le sens indirect (a_5 à a_7).

Pour la construction d'un solide, une enveloppe (ou *shell*) fermée est définie par un ensemble de faces. Ces faces sont orientées de manière à définir l'intérieur et l'extérieur du solide. La figure 3.3 schématise ce que serait le modèle B-Rep d'un cube. On peut constater l'orientation de la normale de chaque face (vecteur normal) vers l'extérieur du volume. Cette orientation est une information qui renseigne sur la notion d'intérieur et d'extérieur du solide. Dans la pratique, selon le plongement géométrique des faces, il n'est plus question de conserver une unique normale au niveau de la face, étant donné qu'elle peut changer en tout point de la face. On conserve alors une procédure de construction de l'information d'orientation de la face en fonction du vecteur normal.

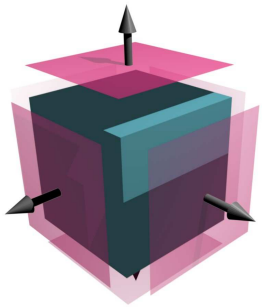


FIG. 3.3 – Représentation B-Rep du cube.

3.1.2 Validité d'un modèle B-Rep

La complexité de prototypage d'un modèle B-Rep peut croître très rapidement avec le nombre d'entités géométriques mises en œuvre. Aussi, des méthodes permettant de vérifier la validité d'un modèle sont indispensables pour assurer sa cohérence. Appliquée à un modèle B-Rep, la formule d'Euler (3.1) permet de vérifier la cohérence du nombre d'entités géométriques de dimensions différentes qui composent un solide :

$$F - A + S = 2 \cdot (C - H) + R \tag{3.1}$$

avec

- S le nombre de sommets (de noeuds) ;
- A le nombre d'arêtes (de courbes de bords) ;
- F le nombre de faces ;
- C le nombre de composantes connexes ;
- H le nombre d'anses ;
- R le nombre de trous dans les faces.

Par exemple, pour un cube on a $S = 8$, $A = 12$, $F = 6$, $C = 1$, $H = 0$, $R = 0$. D'où $F - A + S = 2$ et $2 \cdot (C - H) + R = 2$.

Bien entendu, cette condition n'assure pas que le modèle soit valide mais elle permet de signaler s'il ne l'est pas. Ce test de validité est souvent utilisé pour vérifier des modèles géométriques interprétés et importés à partir de fichiers. Par exemple, lors de l'interprétation d'un modèle décrit par un fichier au format de la norme STEP [Bou95] créé par une autre application, rien n'assure que le modèle soit valide.

A ceci s'ajoutent les incertitudes numériques inhérentes aux modèles B-Rep : dans la pratique, par exemple dans le cas du format d'échange STEP introduit en annexe C, on s'aperçoit que les entités correspondant à des segments sont localisées dans l'espace grâce à des références vers des entités de type point (le plus souvent deux points). Dans un espace à trois dimensions, ces points sont des listes de trois nombres flottants représentés sous la forme $mantisse \times 2^{exposant}$, où la mantisse est représentée par un nombre binaire à p bits. D'une part, il existe une borne sur le plus grand et le plus petit nombre pouvant être représentés ; d'autre part, les nombres réels ne peuvent pas être représentés exactement par un nombre fini de bits. Ainsi, une incertitude est associée à chaque com-

posante d'un point. On s'aperçoit alors qu'un point, a priori clairement identifié par ses trois composantes (x, y, z), se trouve être une entité impossible à localiser exactement dans l'espace. Qu'en est-il de l'appartenance d'un point à un segment ?

Pour résoudre ce problème, il est généralement associé une incertitude numérique ε à un modèle (certains modèles proposent même plusieurs incertitudes), et dans le meilleur des cas, les calculs géométriques peuvent être réalisés en géométrie floue. Ainsi, à un point est associé une boule dont le rayon est fonction de ε , et deux points sont considérés confondus si l'intersection de leur boule n'est pas l'ensemble vide. Ces incertitudes se propagent alors aux segments, aux surfaces et d'une manière générale à toutes les autres entités géométriques du modèle. Concrètement, il est considéré qu'un point appartient à un segment s'il est à une distance suffisamment faible de celui-ci, le seuil de décision étant fonction de ε.

On s'aperçoit alors, qu'une des difficultés liées à l'interprétation des géométries floues, peut être de retrouver la consistance du modèle dans le cas où deux arêtes voisines (au sens topologique) n'auraient pas de points confondus (au sens géométrique), suite à la propagation des incertitudes sur la localisation des points dans l'espace. Cette difficulté se généralise aussi pour toutes les autres entités géométriques du modèle.

3.1.3 Synthèse

La modélisation B-Rep est une représentation de solide fondamentale pour l'ingénierie. Elle offre les supports nécessaires aux cotations. De par sa structure proche d'un maillage (voir paragraphe 2.1.2), un modèle B-Rep est bien adapté au calcul scientifique. Cependant, il est moins bien adapté aux modifications de sous-ensembles de la géométrie[1] qu'un modèle CSG (présenté au paragraphe 3.2).

Les modèles B-Rep nécessitent une connaissance a priori de la structure des objets à modéliser et ne se prêtent pas facilement au prototypage rapide.

3.2 Modèle CSG

3.2.1 Principes élémentaires

Les modèles CSG (Constructive Solid Geometry) ou par historique de construction, proposent une approche différente des modèles B-Rep, tant sur la structure que sur le prototypage : des formes de base sont assemblées par des opérateurs ensemblistes de type union, intersection et différence (voir figure 3.4).

Le modèle est le résultat d'une expression mathématique qui est traduite en un arbre CSG. Dans [CM07], Cazier et Minich définissent un arbre CSG comme étant *un arbre dont les feuilles contiennent des solides élémentaires, ou primitives, et dont les nœuds*

[1] il est cependant possible de mettre en place des méthodes appliquées à des déformations spécifiques, comme par exemple le déplacement d'un trou dans une pièce.

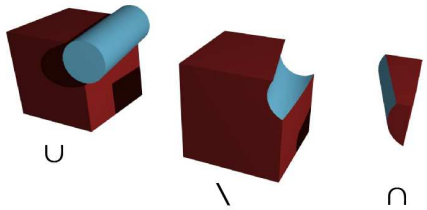


FIG. 3.4 – Exemples d'applications des opérateurs ensemblistes union (∪), intersection (∩) et différence (\) d'un cube et d'un cylindre.

non terminaux contiennent un des opérateurs ensemblistes suivants : union (∪), intersection (∩) et différence (\). Cette définition est enrichie par des opérateurs unaires impactant une branche de l'arbre. Par exemple, les transformations géométriques (rotation, translation, dilatation) sont habituellement caractérisées par des opérateurs unaires. A titre d'exemple la figure 3.5 présente la traduction d'une expression en arbre CSG (en ne tenant pas compte des transformations) et la représentation géométrique de son modèle.

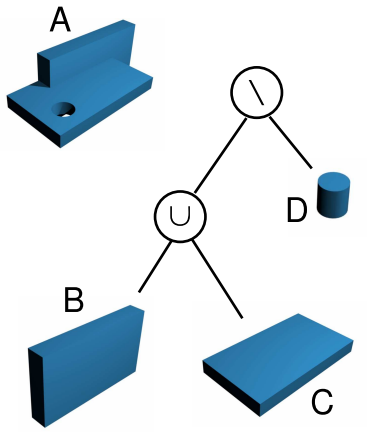


FIG. 3.5 – Un arbre CSG et l'objet A qu'il modélise par l'expression $A = (B \cup C) \setminus D$.

Les primitives peuvent se limiter à une structure qui décrit l'ensemble minimal des paramètres définissant une forme géométrique. Par exemple la boule est définie par quatre paramètres : trois coordonnées d'espace pour le centre et la longueur d'un rayon ; le cube est défini par quatre paramètres : trois coordonnées d'espace et la longueur d'un côté. A ceux-ci viennent s'ajouter les transformations géométriques habituelles qui sont paramétrables : une rotation est définie par un centre (trois coordonnées), un vecteur qui représentre l'axe de rotation (trois composantes) et un angle, soit sept paramètres.

3.2.2 Propriétés

Contrairement à un modèle B-Rep, un arbre CSG ne permet pas d'accéder directement aux informations de cotation, nécessaires aux modeleurs CAO utilisés dans l'industrie. Pour ce faire, il est inévitable de mettre en œuvre des algorithmes géométriques qui calculent ces informations à partir d'un arbre CSG. Ces algorithmes réalisent par exemple des calculs de section de solides décrits par un arbre CSG, des calculs d'intersection

ou des calculs d'englobants parallélépipédiques ou sphériques. S'agissant d'une structure informatique en arbre, la plupart de ces algorithmes sont de type récursif, et ils doivent prendre en compte certaines des propriétés des arbres CSG, énumérées ci-après :

1. **la non unicité** : un solide peut être décrit par un grand nombre d'arbres différents ;
2. **la validité d'un solide CSG** : elle est garantie si les opérateurs sont *régularisés* (terme explicité ci-après) ;
3. **le domaine de représentation** : il est insuffisant dans la mesure où il est difficile de décrire des objets construits par balayage ou décrits par un objet en mouvement (en l'état actuel des opérateurs et des primitives précédemment énoncés) ;
4. **la nature implicite de la représentation CSG** : on sait comment constuire l'objet puisqu'il est défini ainsi, mais on ne sait pas ce qu'il est effectivement.

La non unicité (propriété 1) implique qu'il est possible de modéliser un même objet géométrique avec deux arbres CSG différents. Cette propriété rend délicate les comparaisons d'arbres CSG. Cela ouvre toutefois la possibilité d'optimiser un arbre : on peut imaginer qu'un utilisateur modélise la géométrie d'un objet avec un modeleur CSG, sans ce soucier de l'occupation mémoire ou du coût d'interprétation de son modèle ; puis dans une étape suivante, un algorithme recherche un arbre CSG équivalent plus optimal.

La validité du modèle (propriété 2) est garantie par les opérateurs régularisés. Ces opérateurs, proposés par [Req80], complètent les opérateurs booléens classiques par l'opération de régularisation, basée sur des notions d'intérieur et d'adhérent (notions définies au paragraphe 2.1.2) :

Soit X un ensemble, on dit que X est régulier si $X = \overline{\overset{\circ}{X}}$. L'opération $\overline{\overset{\circ}{X}}$ est appelée régularisation de X.

Notons qu'en dimension 3, l'intérieur d'une face, d'une arête ou d'un point est vide. Ainsi, $\overset{\circ}{X}$ est l'ensemble X auquel sont supprimés les irrégularités et ses bords. L'adhérent permet alors de rajouter les bords. Ainsi, l'opération de régularisation permet de supprimer les irrégularités.

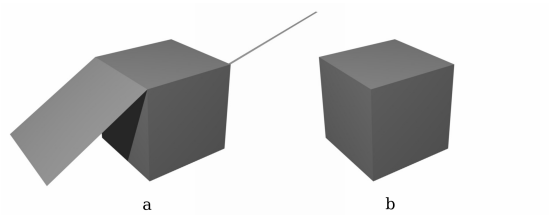


FIG. 3.6 – Illustration de l'opération de régularisation. L'objet a est non régularisé, tandis que b est sa version régularisée.

Concrètement, les opérateurs $\cup$, $\cap$, $\setminus$ sont respectivement remplacés par les opérateurs régularisés $\cup^*$, $\cap^*$, $\setminus^*$, et un opérateur tel que l'union régularisée correspond à l'ex-

pression $A \cup^* B = \overline{A \overset{\circ}{\cup} B}$. Au niveau des objets géométriques, en dimension 3, la figure 3.6 illustre l'intérêt de l'opérateur de régularisation pour valider un solide : seules les *parties* de dimension 3 des objets géométriques persistent après régularisation.

3.2.3 Synthèse

Les modèles CSG sont bien adaptés pour des algorithmes de calcul de type lancer de rayons [Lab07], [POV07]. Ils facilitent la réalisation de modifications[2] et permettent de tenir compte facilement d'un historique des modifications. Par exemple, il est facile de déplacer un trou créé par différence avec un cylindre.

Ces modèles sont moins bien adaptés à la visualisation en temps réel que les modèles B-Rep. Pour être visualisés, les systèmes proposent en général un modèle B-Rep équivalent (bien souvent une triangulation de la géométrie), calculé en fonction de l'arbre CSG. Bien qu'une opération suplémentaire de conversion d'un modèle CSG en B-Rep coûte en temps de calcul, cette possibilté peut permettre de disposer de plusieurs représentations B-Rep d'un même modèle. Ainsi, ces représentations pourraient avoir des incertitudes numériques (voir paragraphe 3.1.2) différentes selon le point de vue d'un observateur. De la même manière, pour réaliser des animations, il est envisageable de modifier des paramètres d'opérateurs unaires (transformations) et de générer successivement un jeu de modèles B-Rep équivalents.

Les modèles CSG sont mal adaptés aux cas de calculs basés sur des maillages tels que ceux présentés au paragraphe 2.1.2, ainsi que pour des outils d'aide à la cotation : les faces et arêtes n'existant pas dans le modèle, leurs caractéristiques doivent être calculées dans un modèle B-Rep équivalent.

3.3 Conversions de modèles

Les deux principaux modèles de CAO, B-Rep et CSG[3], sont des approches complémentaires pour décrire des objets. Aujourd'hui, les outils de CAO existants proposent des modèles hybrides qui gèrent en interne ces deux approches. Aussi, il est fréquent qu'il soit nécessaire de calculer la représentation d'un modèle donné dans un autre modèle.

3.3.1 D'un modèle CSG vers un modèle B-Rep

La conversion d'un modèle CSG en un modèle B-Rep consiste en une évaluation de l'arbre CSG. C'est, en quelque sorte, une fonction plus *naturelle* que sa réciproque. Cette opération est grandement simplifiée si chaque feuille de l'arbre (c'est-à-dire chaque primitive) contient une représentation B-Rep. Il ne reste alors qu'à interpréter l'opérateur qui relie deux primitives, en un équivalent pour des modèles B-Rep. La figure

[2] que ce soit des transformations géométriques ou des modifications de l'arbre CSG qui affectent tout ou partie de la géométrie d'un objet.

[3] d'autres modèles géométriques sont décrits dans la littérature spécialisée. Ces modèles sont souvent adaptés à des géométries particulières. On trouve par exemple des modèles *octree* dont le principe est de décomposer les objets en cubes décrits dans un arbre octal, ou bien des objets construits par balayage (extrusion avec ou sans rotation).

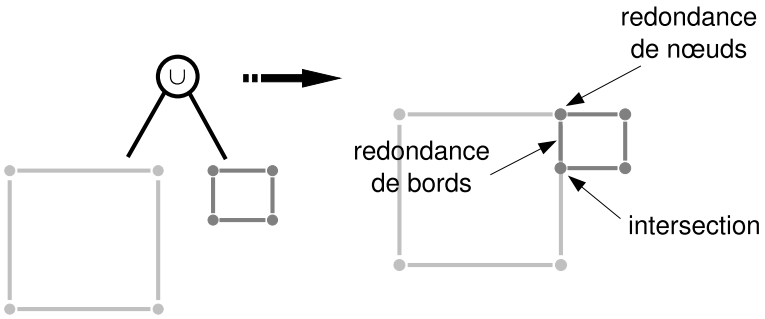


FIG. 3.7 – Conversion d'un modèle CSG vers un modèle B-Rep.

3.7 présente un arbre CSG qui représente une simple union de deux objets rectangulaires. Ces objets sont paramétrés dans des primitives (feuilles de l'arbre) et permettent d'accéder à une représentation B-Rep. Dans ce cas précis, l'opérateur d'union consiste à :

1. Calculer les intersections des nœuds avec chacun des bords des deux objets ;
2. Diviser les bords (entités de dimension 1) pour lesquels il y a eu une, ou plusieurs, intersections ;
3. Eliminer la redondance des noeuds ;
4. Eliminer les bords communs aux deux objets.

Les opérations de calcul d'intersection et d'élimination de redondances des bords supposent des évaluations arithmétiques. Ces évaluations sont sujettes à des erreurs de mantisses (dans le cas des calculs de nombres flottants). Aussi, il est courant que ces calculs soient réalisés à une incertitude préalablement fixée.

3.3.2 D'un modèle B-Rep vers un modèle CSG

Les arbres CSG ont une propriété de non unicité : tout solide peut être décrit par un grand nombre d'arbres différents. Ainsi, le problème qui consiste à proposer un arbre CSG équivalent à un modèle B-Rep est un problème sous-déterminé. Il s'agit d'une problématique proche de celle de l'extraction de caractéristiques de formes (*form feature extraction*) et de la reconnaissance de formes (*form feature recognition*). Le but est d'associer des informations à caractère sémantique à un objet B-Rep qui en est quasiment dépourvu.
Une application de ce type de méthode est la *réingénierie*, qui consiste à prototyper une version n d'un objet, traduit dans un modèle physique tel qu'un maillage. Des études physiques permettent d'optimiser localement le modèle physique (par exemple un maillage de polyhèdre) de l'objet. Il s'agit alors de passer du modèle physique à un modèle CAO fonctionnel [Gar03] (ou de fabrication).

3.4 Modeleur sous contraintes

La modélisation sous contraintes ([JMS07], [MD04]) consiste à enregistrer des contraintes géométriques et non simplement la géométrie qui en découle. Dans l'idéal, la modélisation sous contraintes présente de nombreux intérêts pour les industriels tels que

l'enregistrement des processus de conception, le pilotage de la géométrie par les côtes fonctionnelles, la prise en compte des lois de dimensionnement, le couplage avec des variables de conception et la prise en compte de composants standards.

Il existe, dans les logiciels actuels, deux approches pour gérer les contraintes géométriques et les lois d'ingénierie. Il s'agit des modeleurs paramétriques et des modeleurs variationnels. Ces deux familles de modeleurs sont en fait des surcouches des modeleurs classique par arbre CSG ou B-Rep.

3.4.1 Modélisation paramétrique

La modélisation paramétrique permet de décrire des objets par divers paramètres, au travers d'une interface. En aval de cette description, les objets paramétrisés sont traduits en modèles géométriques de type B-Rep ou CSG. Comme il a été mentionné plus haut (paragraphe 3.2), quatre paramètres suffisent à décrire une sphère. Pour des objets fonctionnels plus complexes, il est intéressant pour l'utilisateur de disposer de primitives élaborées pour paramétrer un tel objet. Dans le cas d'un clou, il peut être modélisé par un modèle B-Rep ou un arbre CSG (unions d'un cône pour représenter la tête, d'un cylindre pour représenter le corps, et d'un cône pour représenter la terminaison du clou). Toutefois la modélisation paramétrique du clou proposée dans la figure 3.8 permet à l'utilisateur de définir cet objet avec seulement la *connaissance* qu'il s'agit d'un clou et trois paramètres : $L1$, $L2$ et $\varnothing$ pour le diamètre de la tête (sous réserve que le diamètre du corps se déduise de celui de la tête).

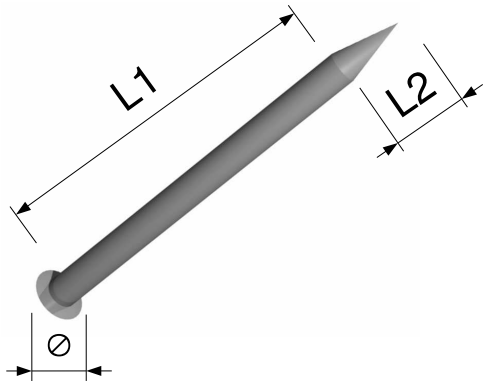


FIG. 3.8 – Représentation d'un clou et de ses paramètres.

Au delà du fait d'avoir une représentation dont le nombre de paramètres a diminué, cette modélisation présente un intérêt fonctionnel : la fonction d'un clou est de lier deux objets. Le clou laisse alors une *trace* dans ces deux objets ; une opération de différence d'un clou avec une primitive représente le fait de *clouer* cet objet. Ainsi une nouvelle couche d'abstraction de la représentation géométrique (les modèles CSG et B-Rep) rapproche le concepteur du modèle réel.

Enfin, un modèle paramétrique permet de réaliser des modifications tout en gardant le même modèle. Supposons que l'on souhaite modéliser une canalisation tubulaire remplie d'eau, et que l'on dispose des primitives tube pour la canalisation et cylindre pour l'eau.

Les paramètres du cylindre sont sa position, sa hauteur et son diamètre. Ceux du tube sont sa position, sa hauteur, son diamètre interne et son diamètre externe. Comme le montre la figure 3.9, le diamètre du cylindre est égal au diamètre du rayon interne du tube.

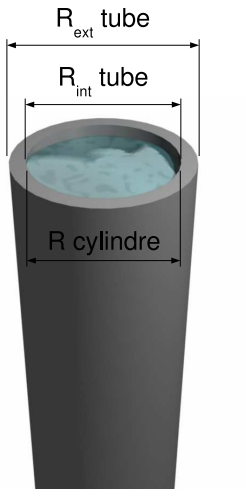


FIG. 3.9 – Modélisation paramétrique d'une canalisation : le rayon R du cylindre d'eau est égal au rayon interne (R_{int}) de la canalisation.

La modélisation paramétrique permet de fixer dans le modèle numérique la valeur du rayon du cylindre à celle du rayon interne de la canalisation. Lors d'une modification de ce rayon interne, le rayon du cylindre est implicitement modifié par le modèle.

Avec un modèle paramétrique, cette dernière fonctionnalité a des limites : l'interprétation dépend de l'ordre de construction du modèle. Lors de l'affectation de la valeur du rayon interne au rayon du cylindre, il est nécessaire que le tube ait été préalablement créé. De même, la modification du rayon interne modifie la valeur du rayon, alors que la réciproque n'est pas permise. C'est la modélisation variationnelle qui va tenter de dépasser ces limites.

3.4.2 Modélisation variationnelle

Dans la modélisation variationnelle, l'utilisateur peut utiliser des outils similaires à la modélisation paramétrique. Cette fois-ci les paramètres des objets du modèle sont des variables d'un système d'équations non linéaires. Ce système est la traduction de contraintes spécifiées lors de la conception du modèle. Par exemple, la contrainte de la figure 3.9 traduite en une équation serait $R = R_{int}$. La différence fondamentale pour le concepteur du modèle, est que le modèle n'est pas dépendant de l'ordre de construction.

L'un des problèmes d'un tel outil de modélisation reste les performances : le concepteur est amené à travailler sur des systèmes CAO interactifs, mais la résolution du système d'équations (résolution de contraintes) peut être coûteuse en temps de calcul. Aujourd'hui, les modeleurs variationnels sont pour la plupart à 2 dimensions, utilisés pour réaliser des esquisses (sketchers) dans des étapes de conceptions préliminaires.

Les modèles de CAO sont alors amenés à évoluer vers des structures disposant d'informations supplémentaires sur les *caractéristiques* des objets modélisés : C'est la modélisation par *entités*.

3.5 Modélisation par entités (modèle orienté *features*)

Les modèles mis en œuvre dans les systèmes de CFAO (Conception et Fabrication Assistée par Ordinateur) actuels sont essentiellement de type géométrique. Or un modèle géométrique seul, n'est pas suffisant pour décrire un objet réel, en un modèle numérique destiné à des études d'ingénierie de conception et de fabrication. D'où la nécessité de compléter ces modèles avec de la connnaissance sur les caractéristiques des objets modélisés.

Ces nouveaux modèles, appelés modèles par entités ([MY98] et [MD04]), par caractéristiques ou encore orientés *features*, sont des imbrications de groupements de caractéristiques. Ces groupements sont les atomes de modélisation appelés *entités*.

Dans les chapitres suivants, nous préférons utiliser le terme *modélisation par features*, ou *modèle orienté features*, les termes *caractéristique* et *entité* étant utilisés par ailleurs.

3.5.1 Définition d'une entité

L'entité apparaît dans les années 80 comme le point commun entre les modèles de description des pièces et les modèles de préparation à la fabrication. A cette époque ce sont principalement des entités de formes dont il est question, et donc des entités nécessairement géométriques. Dans les années 90, une première définition de l'entité est donnée par le groupe GAMA [GAM98] : *Une entité d'usinage est une forme géométrique et un ensemble de spécifications pour lesquelles un processus d'usinage est connu, ce processus est quasi indépendant des processus des autres entités.*

On voit nettement apparaître dans cette définition la volonté d'un modèle commun aux métiers de conception et de fabrication, où la géométrie est fédératrice.

Aujourd'hui, le spectre des métiers qui utilisent des outils de CFAO (Conception et Fabrication Assistées par Ordinateur) s'est élargi avec l'analyse et le calcul et une nouvelle définition plus générale de l'entité a été adoptée :

Un groupement sémantique (atome de modélisation) caractérisé par un ensemble de paramètres, utilisé pour décrire un objet indécomposable utilisé dans le raisonnement relatif à une ou plusieurs activités liées à la conception et à l'utilisation des produits et des systèmes de production.

3.5.2 Utilisation

Parmi les outils existants qui implémentent des modèles orientés entités, on peut citer la norme STEP [Bou95], et le moteur géométrique ACIS [CL01], [ACI00] dont les modèles de données sont proches, basés sur des notions d'*entités* et d'*attributs*[4] (caractéristiques de métiers) qui ajoutent de la sémantique aux modèles. En soi, une *entité* est une notion abstraite, qui peut être concrétisée par des spécialisations géométriques (des points, des courbes, des surfaces, des transformations, ...), par des spécialisations topologiques (des faces, des bords, des enveloppes (*shell*), ...) ou en *attribut*. Ces *attributs* permettent d'associer des caractéristiques à des *entités*, il en existe trois types :
- des *attributs* qui représentent des données simples, telles qu'une couleur ou un matériau ;
- des *attributs* plus complexes, qui font référence à d'autre *entités* ;
- des *attributs* qui associent des données spécifiques à un métier, telles que des paramètres physiques.

Ces modèles orientés entités permettent donc de décrire des objets géométriques avec des *entités* spécialisées dans la géométrie (plongements) et d'autres dans la topologie. A celles-ci peuvent s'ajouter des caractéristiques métiers au travers des *attributs*. Ainsi, un objet est décrit par une base commune à chacun des métiers. Par la suite, il pourra être extrait des vues différentes d'un même objet, spécifiques à chacun des métiers.

3.5.3 Synthèse

Une entité est alors vue comme un modèle géométrique hybride, ayant différentes caractéristiques métier. Ces caractéristiques peuvent être la nomination du matériau et ses propriétés, la fonction technologique de l'objet, ou les moyens de fabrication...

L'objectif de la modélisation par entité étant d'améliorer la communication entre les personnes intervenant sur un produit tout au long de son cycle de vie. Ce concept paraît s'appliquer naturellement à la problématique du contexte de la thèse : le couplage de disciplines de physique et l'évolution des dispositifs tout au long des expériences d'irradiations.

Dans le domaine du nucléaire, la politique d'évolution des outils d'analyses numériques s'oriente clairement vers le partage des compétences des différents métiers, au travers de la plateforme SALOME. Tenant compte de ce contexte, il a été choisi d'intégrer les développements réalisés durant la thèse dans SALOME. Et, afin de bénéficier, au mieux, des apports techniques de cette plateforme, le chapitre 4 présente son environnement.

[4] Cette notion d'attribut n'a pas de réel rapport avec la notion, en programmation orientée objet, d'attribut d'une instance.

Chapitre 4

Environnement SALOME

De nos jours, les produits technologiques, qu'il s'agisse de véhicules, d'équipements électriques, de structures ou de réacteurs nucléaires, sont de plus en plus complexes à réaliser. Cela est dû aux exigences en terme de performance, de qualité et de coût. Les étapes de fabrication et de développement demandent des simulations de plus en plus fines du comportement de ces produits.

Ces dernières années, les outils de calculs ont bénéficié des avancées induites par l'augmentation des puissances de calcul des ordinateurs et l'optimisation des méthodes de calcul. Cependant dans le domaine de la physique des réacteurs nucléaires, mis à part le superviseur ISAS[1] ([GT96]), peu de méthodes de couplage ont été proposées pour améliorer les précisions des modélisations actuelles.

Le projet SALOME[2] définit et réalise une architecture logicielle à base de composants réutilisables pour construire une plate-forme générique de liaison CAO-CALCUL, dédiée à la simulation numérique. La plate-forme SALOME, est un moyen de favoriser le couplage des codes, la réutilisation de portions de codes, l'interopérabilité entre codes de simulation et l'intégration de logiciels de modélisation CAO dans le domaine de la simulation numérique.

Les développements en cours de la nouvelle génération de codes de calcul du CEA et de EDF[3] sont réalisés avec la volonté d'intégrer ces futurs outils de simulation dans SALOME, et ainsi de favoriser les initiatives de modélisations multiphysique et multiéchelle.

Aujourd'hui, les développements de la plate-forme SALOME arrivent à maturité. Des modélisations intégrées à SALOME commencent à voir le jour ([SCCM03], [nur07]), et des outils d'aide à l'intégration de nouveaux composants, tels que XDATA détaillé dans le paragraphe 4.4, sont en cours d'élaboration.

[1] ISAS : Integrated Safety Analysis System.

[2] SALOME : Simulation numérique par Architecture Logicielle en Open source et à Méthodologie d'Evolution.

[3] DESCARTES pour la Neutronique, NEPTUNE pour la Thermohydraulique, PLEIADES pour le combustible, ALLIANCES pour le stockage-entreposage et SINERGY pour les matériaux.

Le projet SALOME offre un cadre privilégié pour le développement d'une plate-forme multiphysique de modélisation des dispositifs expérimentaux.

Dans le domaine du nucléaire, le groupe de travail PAL[4] s'est efforcé de proposer une architecture logcielle, intégrée à SALOME et commune aux différentes disciplines, qui soit le support des simulations multiphysiques appliquées aux réacteurs de puissance. L'originalité du travail de thèse est de mettre en œuvre une architecture similaire, et d'y apporter une généralisation pour l'appliquer aux réacteurs de recherche et à leurs expériences d'irradiation.

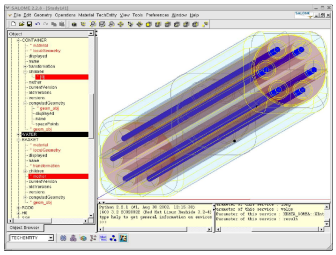
Ce chapitre traite, dans un premier temps, de généralités sur la plate-forme SALOME. Cette plate-forme propose un format d'échange de données : le format MED. On trouve dans le second paragraphe, un aperçu des principales notions du format MED.
SALOME est développé selon une architecture CORBA qui permet d'y intégrer des développements dans des *composants*, présentés au paragraphe 4.3, c'est à dire des fonctionnalités spécifiques à un métier. Pour faciliter cette intégration, un ensemble d'outils, XDATA, a été développé, le paragraphe 4.4 en propose un aperçu. Ces outils ont permis de mettre en œuvre le modèle de données d'Objets Technologiques du groupe PAL, présentés au paragraphe 4.5, dont les développements réalisés durant la thèse se sont inspirés.

Comme nous le verrons dans la suite de ce document, les fonctionnalités de ces différents outils ne sont pas suffisantes pour nos besoins spécifiques, mais elles sont à la base des développements réalisés durant la thèse.

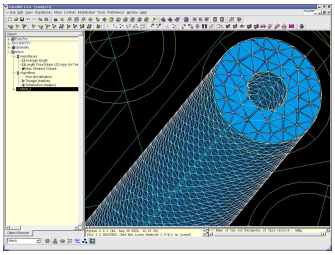
4.1 La plate-forme SALOME

La plate-forme SALOME [Cas07b] est un logiciel libre développé par de nombreux partenaires, dont le CEA et EDF, basé sur la technologie Open Cascade [Cas07a]. Elle met à disposition des outils de pré/post-traitement et de visualisation d'informations relatives à des géométries, des maillages, des champs de valeurs dédiés à la simulation numérique. Ces outils sont accessibles à travers différents composants(figure 4.1). A titre d'exemple, le composant GEOM (composant au sens CORBA, voir paragraphe 4.3) est utilisé pour définir la géométrie d'un objet (figure 4.1(a)) tandis que le composant SMESH (figure 4.1(b)) est utilisé pour réaliser le maillage de la géométrie de ce même objet. Le composant VISU (figure 4.1(c)) permet quant à lui d'obtenir une représentation graphique de divers champs de valeurs sur le maillage. SALOME et ses modules sont utilisables soit à partir de l'interface graphique, la souris étant alors le principal outil d'interaction avec le système ; soit au moyen de script en langage PYTHON.

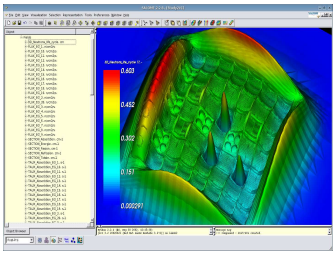
[4] PAL : Projet Architecture Logicielle
Le projet de co-développement PAL a pour objectif la conception et le développement de l'environnement d'utilisation des plates-formes de modélisation et d'assurer la cohérence des architectures de ces plates-formes. Le projet est structuré autour de trois axes principaux :
- La construction et l'évolution de la Plate-forme d'Accueil Logiciels PAL/SALOME, basée sur la plate-forme SALOME ;
- Le support au développement des plates-formes applicatives ;
- L'industrialisation, le déploiement et le support de la plate-forme PAL/SALOME.



(a) Composant GEOM (CAO)



(b) Composant SMESH (Maillage)



(c) Composant VISU (Post-traitement)

FIG. 4.1 – Présentation des principaux composants de la plate-forme SALOME.

La plate-forme SALOME est livrée avec de nombreux composants. Nous ne présenterons ici que les composants utilisés dans le cadre des travaux de thèse. Le lecteur pourra se reporter au site officiel [Cas07b] de SALOME pour plus de précisions sur la plate-forme.

- Le composant *GEOM* est basé sur la géométrie développée par Open Cascade [Cas07a]. Il procure des fonctionnalités de CAO et permet donc de réaliser des calculs géométriques (Technologie Open Cascade). Par ailleurs, il permet d'exporter et d'importer des géométries aux formats STEP (ISO 10303 AP 203/214) [Bou95] et IGES [IGE].
- *SMESH* permet de réaliser des maillages en utilisant différents algorithmes ou en interfaçant le module à un mailleur externe. Il s'appuie sur les géométries du composant GEOM. Les maillages réalisés sont orientés pour des méthodes de résolution éléments finis et différences finies. Ils sont construits à partir d'hypothèses 1D puis 2D puis 3D (pour un maillage volumique) et d'algorithmes de maillage (mailleur). Ces maillages sont convertis et enregistrés au format MED (*M*odèle d'*É*change de *D*onnées) [EC04] fichier ou MED mémoire (voir paragraphe 4.2).
- *VISU* est un composant de visualisation scientifique de maillages et de champs enregistrés au format MED.

Au cours de ce travail, les versions de SALOME ont évolué de la version 2.2.2 à la version 3.2.6. Les développements effectués pendant la thèse ont suivi ces différentes versions. Une des difficultés de la thèse a été de suivre ces développements et que la plate-forme réalisée pendant la thèse reste compatible avec ces nombreuses versions. Il a notamment été nécessaire de vérifier pour chaque nouvelle version, la non régression des fonctionnalités mises en œuvre.

4.2 Le format MED

Le format MED [EC04] a été développé dans le but de fournir un *M*odèle d'*É*change de *D*onnées commun aux simulations numériques multiphysiques. Il s'applique plus particulièrement aux codes de calcul éléments finis et différences finies. Dans l'environnement SALOME, le format MED est présent sous deux formes. Une forme mémoire (MED-mémoire), qui permet d'échanger des données (des maillages et des champs de valeurs) entre différents modules sans passer par une écriture sur disque. Une forme fichier (MED-fichier) qui permet de sauvegarder des maillages et des champs de valeurs sur le disque. Le volume de données que peut contenir un fichier MED peut être important (plusieurs millions de nœuds). Le format de fichier binaire HDF5 développé par le

NCSA[5] a été choisi pour supporter les modèles MED.

Intégrer MED dans un logiciel revient à définir une architecture fonctionnelle composée des trois couches suivantes :

- Une couche basse qui fournit les fonctions d'accès au support d'échange (fichier sur disque) ;
- Une couche intermédiaire qui définit le mode de représentation et d'accès aux données du modèle commun MED ;
- Une couche haute qui gère l'importation et l'exportation des données du modèle commun MED à partir ou vers les structures de données propres à chaque code de calcul.

La couche basse et la couche intermédiaire font partie de la bibliothèque MED-fichier V2.2. Les développements permettant d'intégrer MED dans un logiciel se limitent à la couche haute. Néanmoins, il est important de vérifier la compatibilité d'un fichier enregistré au format MED avec la librairie MED-fichier disponible, ainsi qu'avec la librairie HDF5. En effet, il existe différentes versions de fichier MED qui peuvent se trouver incompatibles avec les librairies disponibles.

La plate-forme SALOME implémente la couche haute de MED et propose ainsi un ensemble de services permettant d'utiliser ce format. Ces services sont mis à disposition à travers un module accessible en C++ et en PYTHON. Dans le travail de thèse, nous avons choisi d'utiliser le format MED au travers de ce module. Ainsi les développements réalisés se trouvent affranchis de toutes incompatibilités vis-à-vis des versions de MED-fichier et HDF5, celles-ci étant gérées par SALOME.

4.3 Composant SALOME

Un composant SALOME est un composant CORBA. C'est une entité logicielle possédant les caractéristiques et les comportements suivants (selon [Taj03]) :
- Physiquement identifiable, c'est à dire qu'un composant n'est pas un concept ou un *pattern* de conception mais un véritable morceau de code binaire, directement exécutable - ou interprétable - après déploiement ;
- Composable, un composant doit pouvoir être mis en relation avec d'autres composants : il expose une ou plusieurs interfaces qui agissent comme des canaux de communication entre composants ;
- Un composant interagit avec les autres composants uniquement au travers de ses interfaces ;
- Un composant est intégré dans une architecture applicative par un processus de déploiement. Ce processus de déploiement est dépendant de la plate-forme matérielle sous-tendant l'architecture ;

Dans une session SALOME (*i.e* lorsque SALOME est lancée), les accès aux composants sont réalisés au travers du *cycle de vie CORBA*. Ce service de `salome` permet entre autres d'accéder à une référence d'un composant, qui peut être installé physiquement sur une machine distante. Par exemple, les lignes de codes suivantes permettent d'accéder

[5] NCSA : National Center for Supercomputing Applications

au composant GEOM (instance g), qu'il soit installé localement ou bien sur une machine distante.

```
import salome
lcc = salome.LifeCycleCORBA()
g = lcc.FindOrLoadComponent('FactoryServer','GEOM)
```

L'instance g dispose de méthodes, qui sont les services du composant GEOM.

Hors d'une session SALOME, les fonctionnalités d'un composant ne sont accessibles que localement au travers d'un module python[6].

Un des avantages de cette architecture logicielle, est de distribuer des fonctionnalités de composants au travers d'un réseau, et de favoriser les échanges de données entre composants en passant nécessairement par des formalismes clairement définis : les échanges entre composants sont réalisés au travers d'appels de services (des méthodes et des procédures) dont les arguments, en entrée comme en sortie, sont clairement identifiés.

Il faut garder à l'esprit que ces outils sont développés pour intégrer des outils de physique. Ces outils, entre autres les codes de calcul, sont eux-même développés par et pour les physiciens. Ainsi, il est important que les difficultés d'ordre informatique, dues à l'architecture de SALOME, soient le plus transparentes possible à cette communauté. C'est dans cette optique qu'à été mis en œuvre XDATA.

4.4 XDATA

L'intérêt de SALOME pour les travaux de thèse est de disposer de composants logiciels (CAO, mailleurs, ...) déjà existants, de formats d'échange et d'un environnement graphique (IHM), distribué (CORBA) et dédié au calcul scientifique. Pour bénéficier de cet environnement, une application doit être développée sous la forme d'un composant de SALOME.

Les outils XDATA permettent de développer un ensemble de classes informatiques et de les intégrer automatiquement dans un composant SALOME. On s'affranchit alors des difficultés d'ordre technique pour l'intégration dans SALOME (interfaces IDL pour CORBA, serveur de noms, ...). Il est donc nécessaire d'étudier en détail ces outils pour réaliser l'intégration des développements de la plate-forme de modélisation dans SALOME.

La définition complète des objets XDATA est décrite dans [Ada04]. L'annexe D, présente l'utilisation de XDATA ainsi que quelques exemples de création d'objets XDATA et leur intégration dans la plate-forme SALOME.

La plate-forme SALOME propose des services et des modèles de données traitant des principaux objets manipulés lors des simulations numériques : géométries, maillages,

[6] Il s'agit généralement d'un module réalisé par swig[swi]

champs de valeur. La mise en données d'une application de simulation physique comprend d'autres données plus spécifiques et évolutives, comme les matériaux (caractéristiques physico-chimiques), les conditions aux limites d'un calcul, les données propres à une discipline qui simule le comportement d'un dispositif. XDATA a été spécifiquement développé pour définir des modèles de données évolutifs, facilement intégrables dans SALOME. Les classes XDATA permettent ainsi de simplifier l'intégration d'un modèle de données dans la plate-forme SALOME. Pour le développeur, XDATA se présente sous la forme d'un module PYTHON et d'un ensemble d'utilitaires dédié à la réalisation de composants XDATA pour SALOME. Dans ce cas, ces composants sont développés en PYTHON, et leurs objets sont des instances de classes XDATA (ces objets dérivent de classes génériques XDATA).
Par ailleurs, le langage PYTHON fonctionne par typage dynamique des variables. De ce fait, ce langage admet qu'une même variable référence deux objets de types différents au cours d'une exécution.

4.5 Les Objets Technologiques

Le logiciel SALOME et son architecture CORBA, permettent d'intégrer des composants logiciels dédiés à une application métier. Il a précédemment été présenté des outils, XDATA, qui permettent de faciliter les développements de tels composants. Le groupe de travail PAL (CEA, EDF) a défini et développé un composant avec ces outils, dont les fonctionnalités, explicitées ci-après, se rapprochent des objectifs des travaux de thèse.

4.5.1 Intérêts et principes

L'ensemble précédent (SALOME, MED et XDATA) permet de disposer de formats d'échange de données et d'un outil d'aide à la création d'un modèle de données. Ce modèle de données, commun à chaque discipline, est représenté par le développement des Objets Technologiques. Ces objets sont orientés vers la description des objets rencontrés dans les réacteurs de puissance.

Différentes versions d'Objets Technologiques ont été développées selon les disciplines de physique. Ainsi la thermohydraulique, le comportement du combustible et la neutronique ont développé leurs propres objets technologiques décris dans [GF03],[Bru05] et [PB05] , en prenant soin de conserver des paramètres en commun.

Un Objet Technologique est la description d'une ou de plusieurs géométries, de matériaux et de paramètres de fonctionnement. Ces objets ne traitent ni les options de calcul, ni les modélisations numériques, ni les résultats de calcul et les maillages. Un Objet Technologique (`TechnologicalObject`) est l'association d'une forme (`Shape`), d'un matériau (`Material`) et de paramètres d'état (`StateParameter`). Des Objets Technologiques spécialisés dans la description d'objets spécifiques (par exemple un assemblage, une cuve de réacteur...) sont développés à partir des définitions des classes présentées sur la figure 4.2. Pour des objets plus complexes, il s'agit de l'association de plusieurs objets technologiques plus simples.

Les Objets Technologiques définis par le groupe de travail PAL permettent de décrire

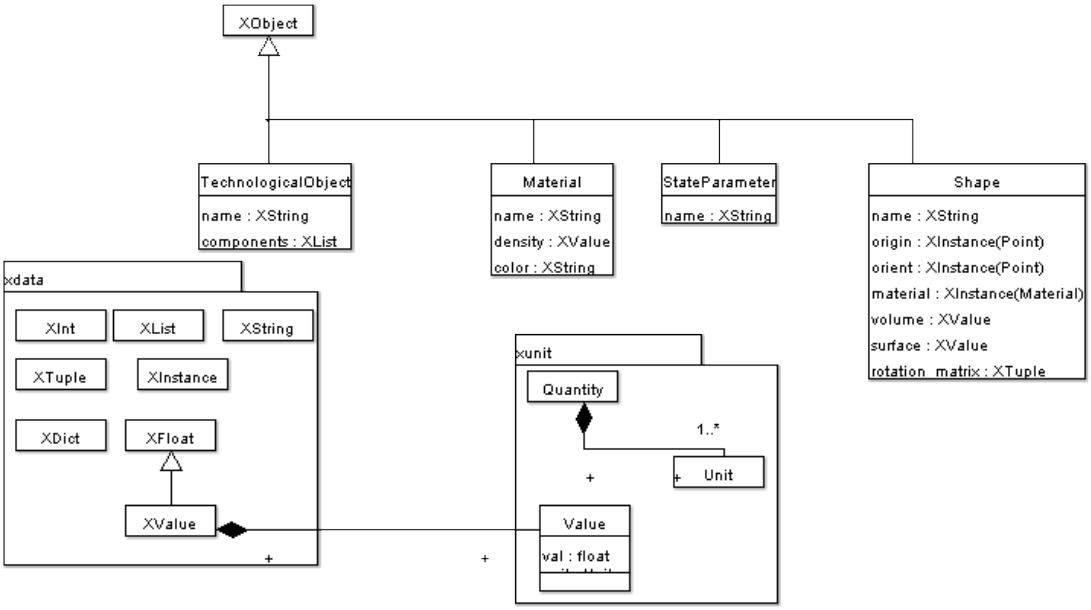


FIG. 4.2 – Objets de base pour la construction d'Objets Technologiques spécifiques (diagramme UML).

un objet tel qu'un réacteur, suivant un certain nombre de règles. De plus, ils ajoutent un contrôle de cohérence sur les types et les domaines de définition des données, car ils dérivent des classes XDATA. Une instance d'un Objet Technologique doit contenir suffisamment d'informations pour permettre de réaliser un jeu de données de neutronique, thermohydraulique et comportement combustible.
Les réflexions du groupe de travail PAL ont abouti à des définitions de classes d'Objets Technologiques représentant les différents composants de réacteurs. La figure 4.3 présente la classe `Assembly` proposée pour le code DESCARTES.

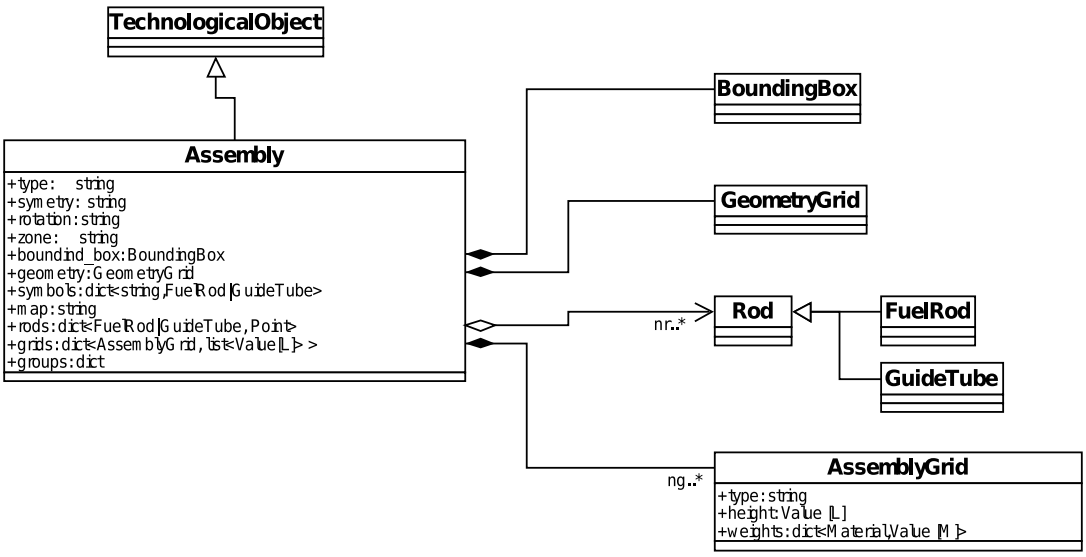
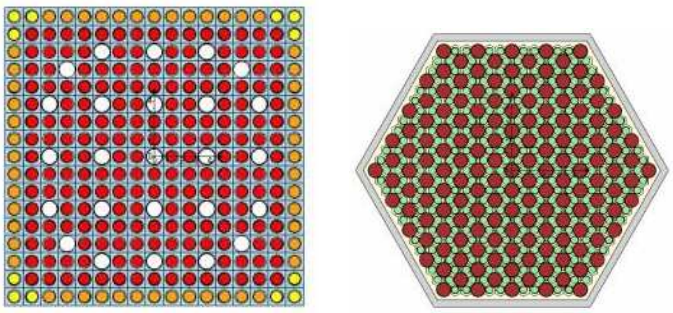


FIG. 4.3 – Diagramme UML d'un assemblage de crayons (Objets Technologiques DESCARTES)

Un assemblage (classe `Assembly`) est un Objet Technologique[7]. Il est composé d'une boîte englobante (`BoundingBox`)et contient des crayons (agrégation de `Rod`). Ces crayons peuvent être des crayons combustibles (`FuelRod`) ou des tubes guides (`GuideTube`) puisqu'un assemblage peut recevoir une grappe de contrôle. Le positionnement des crayons dans l'assemblage est dicté par des règles de placements géométriques (`GeometryGrid`), et les grilles de maintien de ces crayons sont représentées par des `AssemblyGrid`.

Les différentes configurations d'une instance de `AssemblyGrid` permettent de modéliser des assemblages très différents tels que :

- Un assemblage pour réacteur rapide (figure 4.4(b)) avec une bounding_box hexagonale et un positionnement hexagonal des crayons (attribut `geometry`) ;
- Un assemblage pour réacteur thermique (figure 4.4(a)) avec une bounding_box de type boîte et un positionnement régulier des crayons.



(a) Assemblage MOX pour REP (b) Assemblage de conception pour rapide

FIG. 4.4 – Assemblages pouvant être représentés par l'Objet Technologique Assembly.

4.5.2 Exemples de modélisation par Objets Technologiques

Il sera détaillé dans un premier temps, la modélisation de l'assemblage pour réacteur à neutrons rapides représentée par la figure 4.4(b) par les Objets Technologiques. Il sera ensuite présenté la modélisation de l'assemblage REP (figure 4.4(a)). Ces deux modélisations sont réalisées à partir du même Objet Technologique `Assembly` paramétré différemment selon l'assemblage. Un assemblage contient des crayons[8], la figure 4.5 présente le diagramme de classe de `Rod`(crayon) et de ses spécialisations.

Assemblage rapide

Un assemblage est composé d'une boîte englobante (`BoundingBox`), de crayons (`FuelRod`), de grilles de maintien (`AssemblyGrid`) introduisant une perte de charge en thermohydraulique, et d'une grille abstraite définissant les règles de positionnement des crayons dans l'assemblage (`GeometryGrid`). L'assemblage de la figure 4.4(b)) est donc un objet

[7] TechnologicalObject dérivant elle même de XObject (objet de base de XDATA)

[8] Dans le cas des réacteurs à neutrons rapides on parle d'aiguille, pour comparer clairement les deux objets on utilisera le terme crayon.

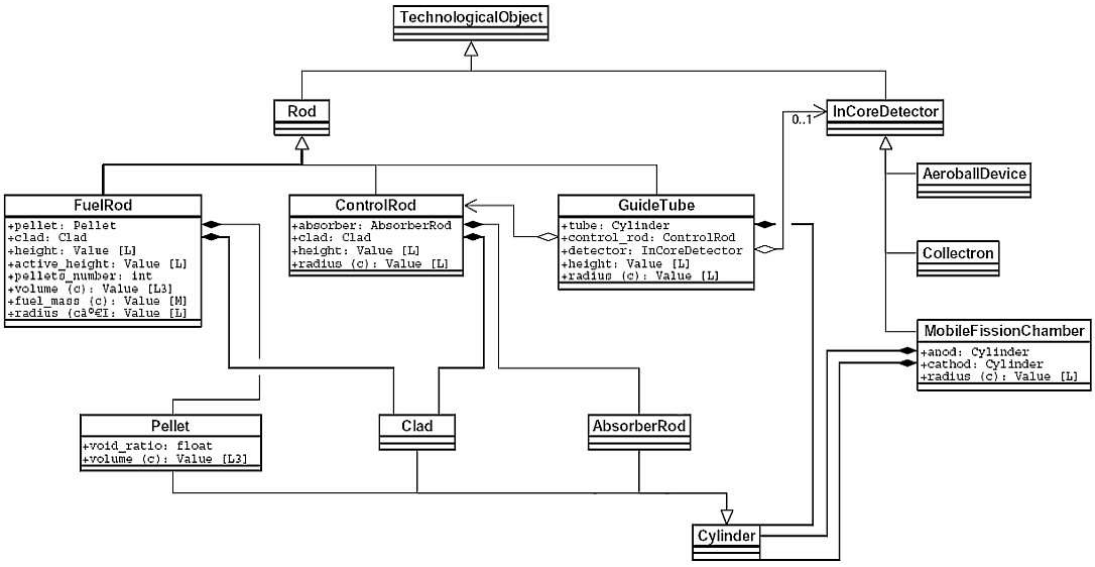
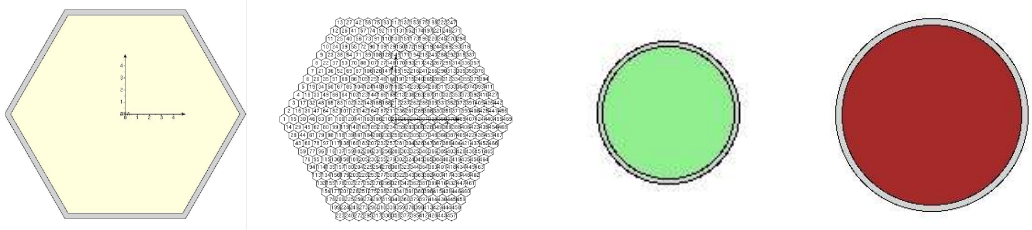


FIG. 4.5 – Diagramme UML d'un crayon (Objets Technologiques DESCARTES) et de ses spécialisations

`Assembly` composé des éléments suivants :

- une boîte englobante hexagonale (figure 4.6(a)) associée à l'attribut `bounding_box` ;
- une grille géométrique décrivant les règles de placement des crayons (figure 4.6(b)) associée à l'attribut `geometry` ;
- des crayons combustibles contenant de l'Uranium (figure 4.6(c)) et des crayons combustible contenant du Plutonium (figure 4.6(d)) contenus par l'attribut `rods` ;



(a) Boite englobante (b) Grille géométrique (c) Crayon combustible Uox (d) Crayon combustible PUox

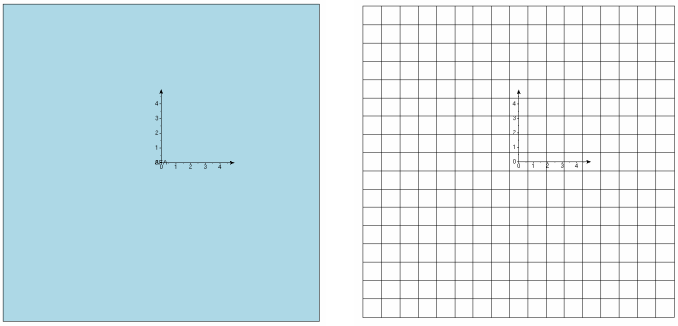
FIG. 4.6 – Eléments composant un Objet Technologique représentant un assemblage pour RNR.

L'attribut `rods` est un dictionnaire permettant de faire correspondre chaque crayon à sa position dans l'espace. Ce dictionnaire n'est pas nécessairement renseigné par l'utilisateur. Il est construit automatiquement à partir d'un plan de placement des crayons (attribut `map`).

Assemblage REP

Dans le cas d'un assemblage pour REP, la boite englobante (attribut `bounding_box`) est cette fois-ci de forme carrée (figure 4.7(a)) et la grille de placement (attribut `geometry`)est

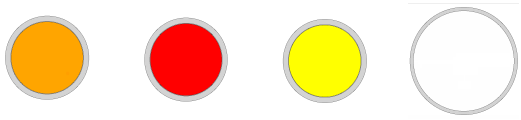
régulière (figure 4.7(b)), composée d'éléments carrés. Le plan de placement des crayons permettant d'obtenir la représentation vue en figure 4.4(a) est cette fois-ci différent de celui de l'assemblage pour réacteurs à neutrons rapides.



(a) Boite englobante (b) Grille géométrique

FIG. 4.7 – Boite englobante et grille géométrique d'un assemblage REP décrit dans un Objet Technologique.

Les crayons (`Rod`) d'un assemblage REP, présentés par la figure 4.8, sont différents de ceux des assemblages RNR. Les assemblages REP nécessitent quatre types de crayons différents (dont un tube guide, figure 4.8(d)) alors que les assemblages RNR sont définis seulement avec deux types de crayons.



(a) Crayon combustible MOX (1) (b) Crayon combustible MOX (2) (c) Crayon combustible MOX (3) (d) Tube Guide

FIG. 4.8 – Eléments de type "crayon" (classe Rod) composant un Objet Technologique représentant un assemblage pour REP.

A partir de ces deux exemples d'assemblages, nous pouvons constater qu'un objet de la classe `Assembly` peut représenter des assemblages très différents, selon la manière dont il est configuré par l'utilisateur. Le modèle que décrit les Objets Technologiques a donc la capacité de représenter deux objets très différents en termes de géométrie et de composition, ayant pour autant la même fonction technologique.

4.6 Discussion sur SALOME et son contexte logiciel

La plate-forme SALOME dispose d'une architecture CORBA permettant d'y intégrer des composants métier. Les récents développements de XDATA, permettent de développer ces composants métier tout en s'affranchissant des interfaces CORBA avec SALOME. Intégrer les développements réalisés durant la thèse dans un composant SALOME permet de disposer des fonctionnalités de SALOME, notamment du format MED

et des services des composants GEOM, SMESH et VISU. De plus, en réalisant les développements avec XDATA, ceux-ci se trouvent directement utilisables par le composant de SUPERVISION de SALOME (ce composant est actuellement en développement). Par ailleurs, par son aspect multiphysique, la plate-forme SALOME est utilisée par diverses communautés scientifiques. Intégrer les développements de cette thèse permet de les rendre disponibles à un grand nombre d'utilisateurs potentiels, et compatibles avec les futures plates-formes de calcul du CEA.

Synthèse de la première partie

Nous avons vu dans cette première partie quels étaient les enjeux repésentés par les réacteurs d'irradiation technologique, ainsi que les géométries des dispositifs que l'on pouvait rencontrer pour étudier le comportement des matériaux sous irradiation. Dans le cadre du développement d'une plate-forme multiphysique permettant de simuler les expérimentations en réacteur, il a été choisi comme premier cas d'application un dispositif d'irradiation de crayons combustibles de type REP.

Le second chapitre a permis de détailler les méthodes de calcul utilisées, ainsi que de présenter de manière analytique le couplage des différents phénomènes physiques rencontrés dans ce type de dispositif. Des hétérogénéités entre les différentes disicplines sont constatées tant au niveau des méthodes de calcul que des codes de calcul, mais aussi au niveau des géométries modélisées.

Les modélisations géométriques envisageables pour la plate-forme ont fait l'objet du troisième chapitre. L'objectif, dans la thèse, étant de mettre au point un modèle paramétrique de données, traitant de géométrie, pour caractériser les dispositifs expérimentaux.

Enfin, des travaux sur le couplage multiphysique dans le cadre des réacteurs de puissance ont été réalisés par le CEA et EDF. Le développement de la plate-forme SALOME et des Objets Technologiques fait notamment partie de cette stratégie d'ensemble.

Les Objets Technologiques développés par le groupe de travail PAL, proposent un modèle paramétrique de données qui permet de décrire des réacteurs de puissance. Il reste à savoir si ce modèle est adaptable aux réacteurs de recherche et à leurs dispositifs expérimentaux. Afin de mieux évaluer ce modèle de données, un exercice appliqué au dispositif CHOUCA est présenté au chapitre 5. La deuxième partie de ce manuscrit de thèse présente les développements proposés, s'appuyant sur les environnements décrits dans cette partie.

Deuxième partie

Mise en œuvre de la plate-forme de modélisation des dispositifs

Dans cette partie, on s'attache à définir un modèle d'architecture pour la plate-forme de modélisation des dispositifs.

Les Objets Technologiques sont proposés comme modèle de données commun à plusieurs disciplines de physique. Ils sont dédiés à la modélisation de réacteurs d'EDF. Les objectifs des Objets Technologiques se rapprochent de ceux prévus pour la plateforme de modélisation des dispositifs expérimentaux développés pendant la thèse. Aussi, a-t-il paru judicieux d'évaluer les capacités des Objets Technologiques à traiter un dispositif expérimental. Pour cela un exercice de modélisation du dispositif CHOUCA, détaillé dans le chapitre 5, a été effectué dans ces travaux.

Cette évaluation permet de proposer des améliorations du modèle d'Objets Technologiques applicables à la modélisation des dispositifs expérimentaux. Ainsi, le modèle de données de la plate-forme de modélisation des dispositifs est défini au chapitre 6.

Le chapitre 7 présente la mise en œuvre d'un tel modèle de données, dans un composant logiciel développé avec XDATA, qui bénéficie des services proposés par SALOME. Le modèle de données développé est inspiré des Objets Technologiques. Il a été baptisé modèle d'Entités Technologiques.

Chapitre 5

Evaluation des Objets Technologiques pour la description de dispositifs expérimentaux

Afin d'évaluer les possibilités offertes par les Objets Technologiques proposés par le projet DESCARTES [LMJL04] et l'utilisation du module XDATA (voir le paragraphe 4.4), une étude préliminaire ([Bon05]) a été menée sur un dispositif de type CHOUCA (voir le paragraphe 1.3.2). Le CHOUCA a été retenu pour ce travail car il possède une géométrie relativement simple, suffisamment représentative des performances de modélisation géométrique attendues pour la plate-forme de simulation. Ce dispositif, utilisé dans OSIRIS, a également fait l'objet d'études pour une intégration dans le réacteur Jules Horowitz ([SBAP04]) .

5.1 Modélisation du CHOUCA avec les Objets Technologiques

A partir des classes définies lors des développements des Objets Technologiques, un ensemble de classes supplémentaires a été développé, pour décrire un dispositif CHOUCA. Il s'agit des classes `Echantillon`, `ChoukSimpleGrid`, `ChoukSimple`, `ChoukSimpleBox` et `CylindricalGrid` présentées sur la figure 5.1. Ces classes permettent de représenter de manière paramétrique la géométrie d'un dispositif CHOUCA, à partir de formes simples (cylindres), de surfaces englobantes et des notions de grilles géométriques pour positionner des échantillons. Le CHOUCA est représenté par la classe `ChoukSimple`.Un CHOUCA contient des échantillons (attribut `echantillon`), chargés selon un plan de chargement (défini par les attributs `map` et `symbol`). Les échantillons sont positionnés selon des règles définies dans cet exercice par la classe `CylindricalGrid` (figure 5.2(c)). La boite englobante du dispositif (figure 5.2(b)), correspondant à la capsule du CHOUCA, est définie par l'attribut `bouding_box` de `ChoukSimple`. Dans ce cas précis, cet attribut référence un objet `ChoukSimpleBox`. Des grilles de maintien des échantillons, représentées par la classe `ChoukSimpleGrid` et référencées par l'attribut `grids`, peuvent être renseignées pour considérer l'impact de la structure sur l'écoulement thermohydraulique[1]. Dans cet exemple ces objets n'ont pas de représentation géométrique.

[1] Par exemple pour déduire des coefficients de pertes de charge

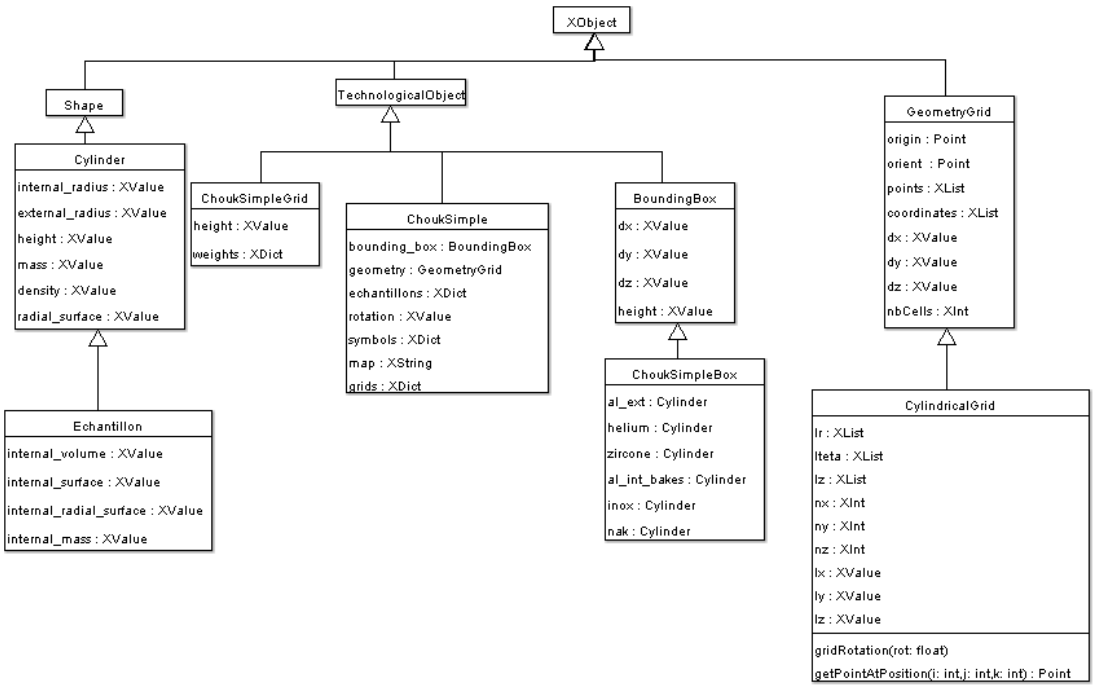
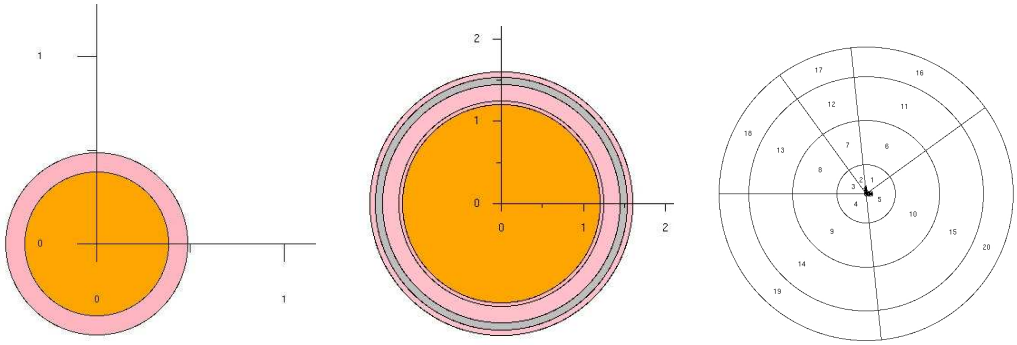


FIG. 5.1 – Diagramme des classes développées pour la modélisation d'un CHOUCA.

La classe `Echantillon` représente un tube contenant un matériau interne cylindrique (figure 5.2(a)). Les mécanismes de typage des objets XDATA permettent, dans le cas de l'échantillon, de contraindre le rayon interne du tube à être inférieur au rayon externe.



(a) Echantillon (b) ChoukSimpleBox (capsule CHOUCA) (c) CylindricalGrid (règles de positionnement)

FIG. 5.2 – Éléments composant un Objet Technologique représentant un CHOUCA.

Il est possible de réaliser des rotations axiales d'un CHOUCA en paramétrant l'attribut `rotation` d'un objet `ChoukSimple`(figure 5.3).

5.2 Discussion sur l'application des Objets Technologiques aux dispositifs expérimentaux

Cette première étude a permis de comprendre le mécanisme des objets XDATA et des Objets Technologiques. Les Objets Technologiques permettent de réaliser des descrip-

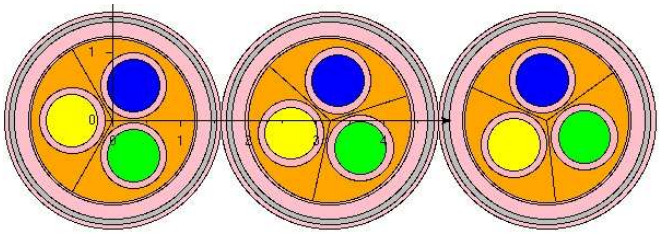


FIG. 5.3 – Rotation paramétrée d'un CHOUCA.

tions paramétriques d'objets. Dans la pratique, les Objets Technologiques créés sont interprétés par les différents codes de calcul. Ce sont ces codes de calcul qui construisent les géométries de ces objets informatiques, et cela à partir des données paramétrées dans les Objets Technologiques. Les Objets Technologiques sont donc exportés vers le modèle de données internes des codes de calcul. Cela suppose que les codes de calcul connaissent la structure des Objets Technologiques traités. Par exemple pour modéliser une pastille combustible, un objet `Pellet` est instancié et paramétré ; cet objet est utilisé comme données d'entrées d'un code de calcul. Ce code de calcul construit, dans son propre modèle de données, a minima un objet géométrique équivalent à ce que représente un objet `Pellet`, dans ce cas un cylindre. Il est donc nécessaire que le code de calcul contienne des procédures spécifiques aux objets `Pellet`. Supposons cette fois-ci que l'objet à modéliser soit un objet `Echantillon`. Cette classe n'a été définie que dans le cas de notre étude, et aucune procédure du code de calcul ne permet de traduire un objet `Echantillon` en objet équivalent de son modèle de données interne : l'objet `Echantillon` n'a pas de sens pour le code de calcul. Pour prendre en compte ces nouveaux objets, il est alors nécessaire de développer des procédures de traduction spécifiques dans chacun des codes de calcul. Cependant, ces travaux sont propres aux développements des codes de calcul, et il n'est pas concevable de réaliser de nouveaux développements spécifiques à chaque dispositif expérimentaux.

La traduction des Objets Technologiques en objets des modèles de données internes des codes de calcul est une opération unilatérale (de l'Objet Technologique vers le modèle interne). C'est-à-dire que les résultats de calculs des codes ne sont pas susceptibles de modifier les paramètres des Objets Technologiques. Ceci implique, dans le cas d'échanges de données entre codes de calcul, que les Objets Technologiques ne peuvent pas être le support bilatéral de communication. Il est alors nécessaire d'élaborer d'autres méthodes pour superviser ces échanges.

Nous pouvons conclure sur le constat suivant : les Objets Technologiques sont un outil puissant pour la description d'objets prédéfinis. Ils sont compatibles avec SALOME et suffisamment généraux pour être multidisciplinaires. Leur modélisation favorise et simplifie les études paramétriques. En effet, il est facile de modifier un paramètre précis tout en conservant la cohérence du modèle.

Les Objets Technologiques ne permettent qu'une relation descendante de la description des objets vers les calculs. La relation ascendante des résultats de calcul vers le modèle ne rentre pas dans le cadre des possibilités des Objets Technologiques, et ce de par leur définition.

La description de la géométrie ne permet pas d'obtenir directement des informations de connexité comme par exemple pour des entités d'un modèle B-REP. Enfin, lors de la description d'objets ayant des formes complexes (autres que des sphères, boîtes, cylindres, ...) il est nécessaire de créer de nouvelles primitives géométriques. En effet, l'ajout d'opérations géométriques[2] n'est pas prévu.

Le placement par grilles favorise les positionnements réguliers d'objets. Tout l'intérêt de ce type de méthode de placement des objets apparait lors d'un très grand nombre d'objets à positionner. Dans le cadre des géométries des dispositifs d'irradiation, le positionnement par grilles peut être limitant puisque les objets, souvent peu nombreux comparés au nombre de crayons des assemblages combustibles, ne sont pas systématiquement positionnés de manière régulière.

La définition du modèle de données commun, appliqué aux dispositifs expérimentaux, pourra donc s'inspirer des Objets Technologiques, mais sera nécessairement plus générale si l'on veut par exemple pouvoir modifier les géométries des dispositifs au cours de l'irradiation. L'objectif du chapitre suivant est de préciser les fonctionnalités du modèle de données pour pouvoir réaliser un chaînage, puis un couplage automatisé dans la plate-forme SALOME.

[2] Par exemple les opérations booléennes.

Chapitre 6

Définition du modèle de données

L'objectif de ce chapitre est de spécifier le modèle de données permettant de décrire les dispositifs d'irradiation. A l'instar des objets technologiques, dont la tendance est de proposer une géométrie fédératrice de vues métier pour chaque application, le postulat de départ est le suivant : la géométrie d'un dispositif sert de base pour la définition du modèle de données, cette géométrie étant la même quelle que soit la discipline de physique abordée. Ainsi, lorsqu'une discipline a pour résultat de modifier cette géométrie, les autres disciplines prennent en compte cette modification par l'intermédiare du modèle de données. L'intérêt apparaît aussi clairement dans le cas d'études paramétriques. Cela impose également un certain nombre de contraintes, en partie dues aux spécificités des codes de calcul et des maillages, que doit respecter le modèle de données.

Le premier paragraphe de ce chapitre détaille les besoins de la plate-forme en terme d'architecture, tenant compte des contraintes liées aux maillages et aux codes de calcul ainsi qu'à la description de la géométrie. Puis la problématique d'échange de grandeurs entre les disciplines est abordée dans le paragraphe 6.2.

6.1 Objectifs et contraintes du modèle de données

6.1.1 Besoins pour les maillages et codes de calcul

Les codes de calcul, présentés au chapitre 2, sont des programmes informatiques permettant de simuler des phénomènes physiques. Il s'agit d'implémentation de méthodes de résolution de problèmes de physique. Un code de calcul est en général développé pour une discipline de physique bien précise. On peut classer ces outils dans deux grandes catégories selon leurs méthodes de résolution des problèmes : les méthodes de résolution déterministes et les méthodes de Monte-Carlo (méthodes de résolution statistiques).

Rappelons qu'un jeu de données est un modèle numérique dont le formalisme est spécifique à un code de calcul, il décrit a minima :

– des discrétisations du domaine d'étude, appliquées à la géométrie[1], à la durée (schémas en temps, pas de Burn up), à des paramètres physiques (sections efficaces multi-

[1] Un maillage est considéré comme une discrétisation d'une géométrie.

groupes, ...) ;
- des caractéristiques sur des matériaux associés à cette géométrie ;
- une méthode de résolution ;
- des hypothèses (modèle physique, simplifications et conditions aux limites)
- des traitements et des requêtes pour obtenir des résultats de calcul.

On constate que ces jeux de données sont souvent les seuls moyens de dialogue avec les codes de calcul. Il peut s'agir aussi bien de simples fichiers de configuration, que de véritables programmes écrits dans des langages spécifiques à un code de calcul (Par exemple le langage GIBIANE pour les codes APOLLO2, CRONOS2 et CAST3M, ou bien le langage LU pour le code ERANOS traitant de la neutronique des réacteurs à neutrons rapides).

Pour les méthodes de calcul déterministes, les méthodes éléments finis et différences finies sont très utilisées en simulation numérique. Les maillages nécessaires à ces simulations sont clairement définis dans [FG99]. Toutefois, dans nos travaux, les simulations neutroniques des coeurs de réacteurs expérimentaux sont réalisées avec une méthode originale : la méthode des caractéristiques ([HAB+05] et [SBd+07]). Cette méthode et les outils de maillage associés (Entre autres le logiciel SILENE [SS93], [Sta96], [Sta97]) permettent de réaliser des simulations sur des maillages plus proches des géométries exactes des objets à modéliser.

Les méthodes de Monte-Carlo ne nécessitent généralement pas de maillage particulier. La géométrie peut-être directement décrite selon les deux représentations B-REP ou CSG[2]. Les résultats de calcul sont traités sur des sous-ensembles de la géométrie.

Un modèle de données multiphysique, permettant de générer, configurer et contrôler des simulations numériques doit présenter des spécificités, propres à l'utilisation des différentes familles de codes de calcul et de maillages. Dans l'idéal, à partir du modèle de données défini pendant la thèse, il doit être possible de configurer, voire de créer, le jeu de données d'une expérience d'irradiation, quelle que soit la discipline de physique traitée et le code de calcul utilisé. Les spécificités de ce modèle de données, concernant ses capacités à communiquer avec des jeux de données de différentes disciplines, sont listées ci-dessous. Elles concernes les besoins relatifs aux discrétisations, aux descriptions des caractéristiques des matériaux, aux méthodes et aux hypothèses de calcul, ainsi qu'aux résultats de calcul. Les besoins en fonctionnalités géométriques sont traités au paragraphe 6.1.2.

Discrétisation

Le modèle d'irradiation de dispositif décrit une géométrie. Au vu de la grande hétérogénéïté des familles de maillages utilisés par les codes de calcul, il paraît préférable d'utiliser les outils de maillages que proposent les codes de calcul et ceux proposés par le composant SMESH de SALOME. Ainsi, le modèle développé ne tiendra pas compte des maillages de calcul. Cependant il se doit d'être en mesure de fournir des représentations géométriques compatibles avec les différents mailleurs.

[2] Pour le code TRIPOLI4, sont utilisés les termes *geométries surfaciques* et *géométries combinatoires*.

Caractéristiques des matériaux

Des matériaux sont attribués aux objets géométriques. De ce fait, le modèle d'irradiation est en mesure de retourner des correspondances entre matériaux et objets géométriques. Les matériaux ont pour rôle d'englober un ensemble de caractéristiques physiques. Ces caractéristiques peuvent évoluer au cours des simulations puisqu'elles peuvent évoluer dans le temps.

Méthodes de résolution et hypothèses

Les méthodes de résolution sont spécifiques aux codes de calcul et aux disciplines de physique abordées. Il en est de même pour les hypothèses de calcul. Pour assurer des fonctionnalités multiphysiques, le modèle d'irradiation des dispositifs doit être, si possible, le plus indépendant des codes de calcul et donc des méthodes de résolution. Il est cependant recommandé que des fonctionnalités permettant de spécifier des conditions aux limites sur des géométries soient mises en place.

Résultats de calculs

Les enchaînements et les couplages de calculs supposent des échanges de résultats de calculs. Ces résultats peuvent correspondre soit à des données globales, et dans ce cas il s'agit d'échanger des listes de données, soit à des champs de données associés à des maillages en espace et en temps des domaines d'études. Il s'agit alors d'échanges de listes de données, cette fois-ci localisées en espace et en temps. Par ailleurs, il est envisageable de considérer les unités des données (une solution technique avait été proposée dans les développements des Objets Technologiques).

Les maillages définis pour les simulations numériques sont des discrétisations de formes géométriques. Les méthodes de discrétisation sont choisies par les physiciens, adaptées au problème traité. Différents maillages d'un même objet sont ainsi réalisés pour modéliser différents phénomènes physiques. Toutefois ces maillages ont pour origine les mêmes formes géométriques, les données relatives à la géométrie d'un objet sont donc communes à chacune des disciplines. Ainsi, un modèle de données multiphysique doit être en mesure de décrire des objets géométriques.

6.1.2 Besoins en géométrie

Le modèle géométrique souhaité pour le modèle générique d'expérience d'irradiation doit répondre à des critères provenant des spécificités des outils de modélisation des disciplines de physique (les codes de calculs, les méthodes de calcul et les outils de maillages de géométries). Nous devons garder à l'idée que ce modèle générique d'expérience d'irradiation sera utilisé durant différentes étapes du cycle de vie du dispositif.

Dans ce paragraphe, sont traitées les fonctionnalités géométriques souhaitées pour les étapes de construction et de modification des modèles.

Construction

Le modèle géométrique doit être intuitif pour la phase de conception durant laquelle il faut manipuler des entités géométriques, les modifier, appliquer des opérateurs de transformation. Une représentation de la géométrie par un arbre CSG semble être appropriée. Des moyens de visualisation des objets géométriques sont nécessaires pour s'assurer de la cohérence du modèle en cours de conception.

La géométrie varie peu axialement mais il est possible que des éléments soient présents de manière irrégulière à différentes hauteurs. Nous ne pouvons donc pas nous limiter à une géométrie 2D extrudée, mais un véritable modèle géométrique 3D est nécessaire.

Modifications

L'analyse de besoins montre que le modèle d'irradiation des dispositifs doit subir de nombreuses modifications, notamment au niveau de la géométrie.

- Durant la phase de conception, des versions différentes d'un dispositif peuvent être proposées. Ces versions diffèrent soit en tout point du dispositif, soit présentent de légères modifications d'un même objet.

- Dans certains cas, il est nécessaire de réaliser des études paramétriques pour optimiser les conditions d'irradiation (par exemple pour minimiser les gradients de températures). Un modèle géométrique paramétrique simplifie ces études.

- La phase d'exploitation tient compte de l'usure du dispositif et des modifications induites par l'irradiation des échantillons (gonflements, contraintes mécaniques, changements d'état).

- La phase de maintenance d'un dispositif d'irradiation, correspond à l'adaptation et l'utilisation d'un même dispositif pour différentes expériences d'irradiation, qui demandent des modifications locales de celui-ci (ajout d'appareils de mesures, modifications des échantillons, des matériaux...).

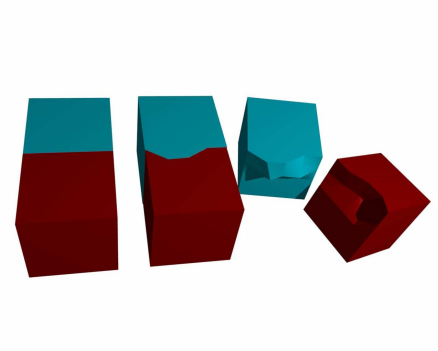


FIG. 6.1 – Exemple de modification des surfaces de deux objets connexes.

– Lors des simulations, certaines disciplines (par exemple lors de calculs thermomécaniques) sont amenées à modifier le maillage de calcul en déplaçant les nœuds de celui-ci. Dans la pratique, un résultat sous forme de champ de vecteurs associé au maillage de calcul permet d'effectuer des translations de chaque noeud. Cela est interprété comme une modification de la géométrie du dispositif.
Ces modifications se caractérisent pas des déformations des surfaces des géométries des objets[3] : par exemple des surfaces planes vont tendre à être gauches. De plus, lorsque deux objets (ou plus) sont connexes, la modification de la surface d'un premier objet influe sur la surface connexe d'un second comme il l'est illustré figure 6.1. Ces modifications peuvent intervenir localement sur la surface d'un objet. Pour tenir compte de telles déformations, un modèle B-REP est généralement utilisé.

6.1.3 Discussions sur les développements de la thèse

Le travail de thèse doit permettre de développer une plate-forme pour la modélisation de dispositifs expérimentaux (voir figure 6.2) durant les phases :

– de développement de modélisations multiphysiques ; plusieurs modélisations physiques sont réalisées et sauvegardées selon différentes versions. Par exemple, il peut exister différentes versions de modèles physiques traitant d'un même phénomène, mais avec des hypothèses, des méthodes de calcul ou des maillages différents. Il est alors possible de comparer ces modèles lors d'étapes de validation et de qualification.

– de développement d'un dispositif ; un ensemble de modélisations multiphysiques d'une expérience d'irradiation est développé et figé. Des études paramétriques sont réalisées à partir de cette modélisation, agissant sur des variables de conception qui impactent les géométries ou les caractéristiques des objets. Il peut en être déduit des modifications du dispositif et certaines de ces variables de conception peuvent être figées. Un mécanisme de gestion de version permettrait alors de tenir compte de ces modifications.

– d'exploitation, la modélisation de l'expérience d'irradiation est préalablement développée. Des paramètres du modèle sont modifiés pour vérifier et/ou prévoir des comportements. Les modèles sont remplacés et/ou recalés par des bases de données expérimentales.

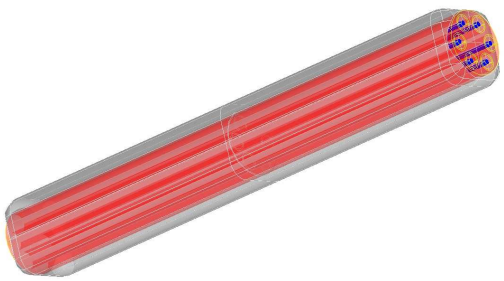


FIG. 6.2 – Exemple d'un dispositif d'irradiation.

[3] Dans le cas d'objet géométrique de dimension 3.

Alors que pour faciliter la phase de conception d'un dispositif donné, une modélisation CSG semble plus adaptée, les deux modèles géométriques CSG et B-REP restent néanmoins nécessaires pour décrire les objets géométriques. C'est pourquoi, pour décrire les objets du modèle de données proposé par la thèse, il est choisi une solution sur la base d'un modèle géométrique hybride (B-Rep/CSG).

Pour cela, les services du composant GEOM de SALOME permettent de construire des objets à partir d'objets géométriques élémentaires : des points, des bords, des faces, des volumes, ou bien à partir de primitives géométriques. Par exemple pour construite un cube de côté unité en PYTHON, on pourra utiliser un script similaire à l'annexe G. Le même cube peut être construit en utilisant les primitives géométriques à partir des quelques lignes suivantes :

```
import geompy
box = geompy.MakeBoxDXDYDZ(1,1,1)
id_box = geompy.addToStudy(box,'box')
import salome
gg = salome.ImportComponentGUI("GEOM")
gg.CreateAndDisplayGO(id_box)
```

La première ligne permet d'accéder aux fonctionnalités offertes par le composant GEOM, au travers d'un module nommé `geompy`. Les deux lignes suivantes créent un objet (`box`) qui décrit une boîte cubique de côté 1 et un identifiant pour le référencer dans la plateforme SALOME. Les trois lignes suivantes permettent d'afficher cet objet dans l'interface graphique de SALOME.

A partir de la boîte (l'objet `box`) ainsi créée, il est possible de retrouver les entités géométriques ayant permis sa construction grâce à des méthodes (ou *services*) de décomposition proposées par le composant GEOM (telles que `SubShapeAll(...)` du module `geompy`). A titre d'exemple, la ligne suivante permet de retrouver l'ensemble des arêtes du cube `box` :

```
l_edge = geompy.SubShapeAll(box,geompy.ShapeType["EDGE"])
```

La liste `l_edge` contient alors douze objets représentant les douze arêtes du cube.

Des opérations ensemblistes telles que les différences, les unions et les intersections d'objets géométriques sont proposées par les services de GEOM. Il est donc possible de construire des géométries à l'aide des service de GEOM, sur la base d'un modèle B-REP mais aussi d'un modèle CSG.

D'autres méthodes permettent de retrouver des entités géométriques partagées par deux objets géométriques (telles que la méthode `getSharedShapes`). De ce fait, il est possible de reconstruire les informations de connexité entre objets.
Cette fonctionnalité présente un intérêt majeur lorsqu'il s'agit de mettre en œuvre des couplages de simulations réalisées sur des domaines différents. Il est alors nécessaire de pouvoir localiser les interfaces d'échange entre les domaines (ces interfaces d'échange sont en quelque sorte des sous-domaines). Par exemple lors du couplage thermique / thermohydraulique sur des domaines fluide et solide, l'interface commune aux deux domaines est une paroi où se déroulent des échanges thermiques par conduction.

Ainsi, les services du composant GEOM permettent de passer d'une représentation B-REP à une représentation CSG et inversement. Il est de ce fait possible d'utiliser ce composant pour réaliser les calculs géométriques du modèle proposé par la thèse.

En plus des informations géométriques, les simulations physiques nécessitent des connaissances sur les différents matériaux qui caractérisent les domaines géométriques étudiés. Nous proposons donc d'associer à l'information géométrique, une information permettant de caractériser ces matériaux. Toutefois, suite aux travaux du groupe PAL, on s'aperçoit qu'il est difficile de caractériser exactement les différentes propriétés des matériaux, communes aux différentes disciplines abordées. Par exemple, si l'on considère une matrice d'un matériau donné, certaines propriétés caractérisent des comportements microscopiques (sections efficaces microscopiques, structure christalline, ...), d'autres des comportements macroscopiques (sections efficaces macroscopiques, conductivité thermique, viscosité, ...) de la matière, encore d'autres peuvent caractériser son état (température, pression, composition isotopique, ...). Si l'on suppose des problèmes multi-échelles, toutes ces quantités peuvent être aménées à être distribuées sur la matrice selon des lois non-uniformes, et peuvent varier en fonction de divers paramètres (temps, position, ...). Aussi, envisageons-nous d'orienter notre modèle de donnée vers un modèle orienté *Features*, présenté au paragraphe 3.5, qui associe une représentation hybride des géométries et un ensemble de *caractéristiques* associées. En pratique cela correspond à associer à des objets géométriques, le nom d'un matériau et un ensemble d'*attributs* complémentaires, définis par les utilisateurs.

Ce choix technique permet de faire cohabiter les deux modèles géométriques et d'associer à la géométrie des informations concernant les matériaux. Par ailleurs, cette solution a déjà été adoptée, notamment, dans [Lee05] où Sang Hun Lee traite un problème similaire aux travaux de thèse. Il propose d'utiliser un modèle orienté features pour réaliser une interface entre un modèle de CAO et un modèle d'IAO. Son modèle génère des représentations géométriques adaptées à chaque métier.
De plus les modélisations orientées Features ouvrent notre modèle à de nombreuses améliorations, notament pour assurer la cohérence du modèle vis-à-vis du cycle de vie du dispositif. La forme géométrique d'une simple pièce peut être représentée différemment selon une phase de modélisation ou bien une phase de fabrication.

La figure 6.3 illustre ce concept : une même pièce `A` est vue comme un cube auquel à été retiré de la matière (`B`), ou bien comme un parallélépipède auquel a été ajouté de la matière (`C`). Lors d'une modélisation de cette pièce, les résultats d'une étude paramétrique pourraient être amenés à modifier la vue `C`. Ces modifications seraient répercutées par des modifications de l'outil d'usinage de la vue `B`. Des solutions à ce problème, basées sur des modèles orientés Features, permettent de gérer automatiquement des paramètres communs à différentes vues du même objet.

Par exemple dans [SG05], l'auteur cherche à maintenir une cohérence entre les différentes vues d'une même pièce. On voit apparaître l'intérêt de telles fonctionnalités dans le contexte de la thèse : alors que le mécanicien s'intéresse à la géométrie *exacte* d'un assemblage et de ses crayons, le thermohydraulicien s'intéresse au *périmètre mouillé* (c'est à dire la somme des périmètres des objets en contact avec le fluide caloporteur, à une côte axiale donnée). Lors d'une modification de la forme, d'un ajout ou d'un retrait

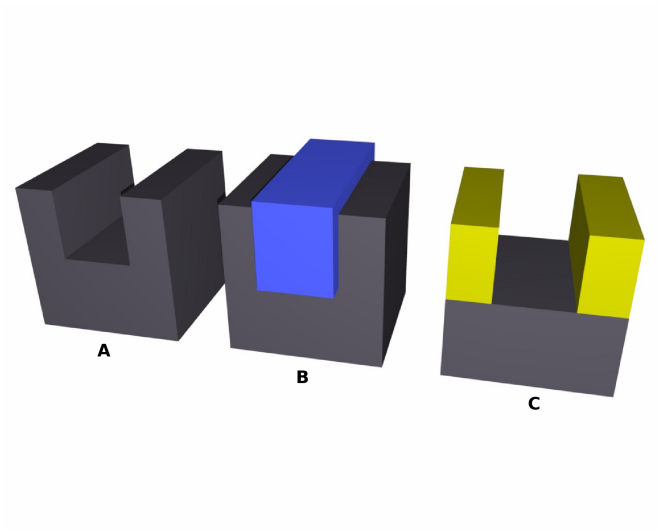


FIG. 6.3 – Différentes vues possibles d'une même pièce (A : La pièce finale, B : La pièce et son outil d'usinage en vue de la fabrication, C : La pièce vue pour une modélisation).

de crayon, la géométrie vue par le mécanicien est fortement impactée et le périmètre mouillé vu par le thermohydraulicien est modifié. Il y a donc une cohérence à respecter entre la vue du mécanicien et celle du thermohydraulicien.

Dans [Li01] une approche inverse au problème est traitée. Après avoir décomposé la géométrie d'un objet en formes caractéristiques, l'auteur reconstruit l'objet à partir d'un arbre de features. Chacune des features de cet arbre peuvent être optimisées lors d'études d'ingénierie, pour répondre à différentes contraintes fonctionnelles. Lors de ces optimisations individuelles, le modèle orienté features permet de garder la cohérence entre les features.
Une application directe de ces travaux serait, à partir du modèle d'un dispositif expérimental, d'optimiser localement la géométrie et les caractéristiques des matériaux de différents objets le composant, en fonction de diverses contraintes. Une étape de reconstruction permet ensuite de regrouper ces objets pour reconstruire le modèle du dispositif dans son ensemble tout en respectant la cohérence des différentes vues.

Cependant, les objets créés par le composant GEOM ne sont pas paramétriques, c'est-à-dire qu'il n'est pas possible de créer dans un premier temps, un cube de côté unité et dans un second temps, de modifier sa taille, sa position ou son orientation. Pour assurer des fonctionnalités paramétriques au modèle d'irradiation des dispositifs, nous proposons de définir de nouveaux objets géométriques. Ces nouveaux objets paramétriques disposent de méthodes permettant la création d'objets GEOM. En substituant ces objets aux objets GEOM, le modèle bénéficie alors des fonctionnalités du composant GEOM à partir d'objets géométriques paramétriques.

En ce qui concerne l'architecture de la plateforme de modélisation des dispositifs, il apparaît que le modèle d'irradiation des dispositifs centralise les données relatives aux simulations, que ce soit le modèle géométrique et les matériaux associés, ou bien les résultats de calcul.

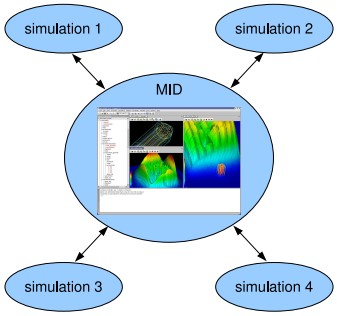


FIG. 6.4 – Échanges de données entre le modèle d'irradiation des dispositifs (MID) et différentes simulations.

Comme le montre la figure 6.4, une telle architecture suppose que des données soient échangées entre le modèle d'irradiation des dispositifs et chaque simulation. Le principal intérêt vient du fait qu'il est alors possible de superviser l'ensemble des échanges entre les disciplines : les échanges inter-disciplines sont permis et transitent nécessairement par le modèle d'irradiation des dispositifs, ce dernier peut ainsi les contrôler (les autoriser, les interdire, les modifier, ...). Il reste donc à proposer des méthodes d'échange de ces données entre le modèle et les simulations.

6.2 Proposition d'une méthode d'échange de grandeurs entre les disciplines de physique

6.2.1 Deux phases d'échanges de données

Nous proposons de réaliser l'échange de données entre les simulations, lors de couplages ou de chaînages, au travers du modèle d'irradiation des dispositifs (figure 6.4). Deux sens d'échanges de données[4] ont été référencés : l'un *descendant* du modèle vers chacune des simulations, l'autre *ascendant* de chacune des simulations vers le modèle.

Lors d'un échange descendant, une simulation est renseignée de l'ensemble des données nécessaires pour effectuer un calcul. De ce fait, ces échanges ne peuvent intervenir qu'en amont d'un calcul. Il s'agit donc d'une phase d'initialisation de la simulation.
Dans ce cadre, les types des données échangées peuvent être des objets CAO[5] ; des identifiants de matériaux (des objets représentant des matériaux, des numéros ou des chaînes de caractères) permettant de pointer sur des caractéristiques propres à la discipline ; des champs de valeurs localisés dans l'espace et dans le temps portés par un maillage ou un sous-maillage du domaine. De plus, une expérience d'irradiation se déroule pendant un certain temps et à un certain moment, l'initialisation d'une simulation peut donc disposer des informations de durée et de temps initial.

Lors d'un échange ascendant, le modèle de données est renseigné des résultats de calcul d'une simulation. Cet échange ne peut donc avoir lieu qu'en aval d'un calcul. Il s'agit d'une phase de validation d'un calcul.

[4] En considérent le sens d'échange de A vers B comme étant le transport des données *connues* de A renseignées à B.

[5] Ou bien des parcours des objets géométriques du modèle de données sous forme de requêtes (*get*).

Les données échangées avec le modèles sont des champs de valeurs localisés dans l'espace et dans le temps, ces données sont donc de type maillage ou des sous-maillages du domaine, renseignés aux instant t et durée Δt donnés. Les post-traitements des disciplines peuvent parcourir le modèle de données pour mettre à jour[6] des informations traitant de la géométrie et des matériaux.

Mis à part qu'il est possible dans le cas d'échanges ascendants de modifier le modèle, les types de données échangées sont identiques dans les deux sens. Puisque le modèle d'irradiation des dispositifs a le rôle d'un intermédiaire entre les simulations, il se doit de disposer d'un moyen d'échange des résultats sur des maillages dans un formalisme commun à toutes les simulations. Deux schémas d'échanges de ces données ont été envisagés : soit au travers d'un maillage de référence commun, soit au travers d'interfaces spécifiques.

6.2.2 Schéma d'échange par maillage de référence

Le modèle d'irradiation de dispositifs échange des données avec diverses simulations. Ces données sont pour certaines contenues dans des formats de maillages d'un sous-domaine du modèle. D'une simulation à l'autre, un même domaine peut être maillé différemment. Une solution d'échange des données entre les disciplines consiste à créer un maillage de référence commun à chaque discipline. Ce maillage est construit initialement par un utilisateur averti qui le considère comme porteur de toute l'information pertinente pour le processus de simulation. Il peut par exemple être l'union des maillages de chaque simulation.

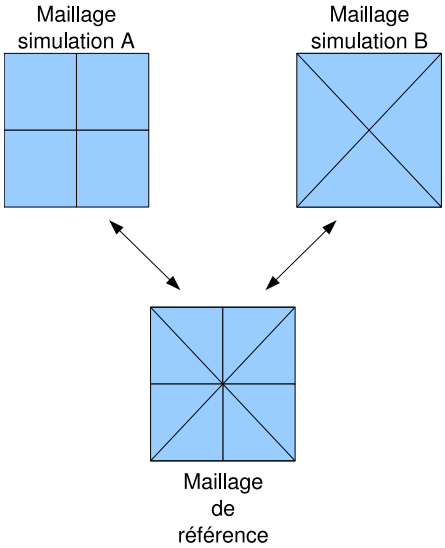


FIG. 6.5 – Principe d'échange de champ par un maillage de référence (union des maillages de chaque simulation).

Dans le cas du maillage de référence illustré par la figure 6.5, à chaque maille de la

[6] Modifier localement le modèle

simulation A correspond une union de mailles du maillage de référence, et il en est de même pour le maillage de la simulation B. Résultant d'une union, ce maillage de référence présente l'avantage de minimiser le nombre d'entités géométriques qui le compose en évitant les rédondances. Par exemple sur la figure 6.5, les noeuds des maillages des simulations sont confondus. De ce fait, le nombre de noeuds du maillage de référence est inférieur à la somme des noeuds des maillages des simulations. Il en est de même pour les bords.

Lors d'un couplage des simulations A et B, dans un premier temps la simulation A réalise un calcul et les résultats sont enregistrés dans le maillage A. Ces résultats sont ensuite projetés sur le maillage de référence. Dans un second temps, la simulation B s'initialise à partir des résultats de A, qui doivent être enregistrées dans le maillage B. Pour ce faire, les résultats de A, précédemment enregistrés dans le maillage de référence, sont projetés sur le maillage B.

Il a été vu au paragraphe 2.1 qu'il existe des nombreux types de maillages, et certaines méthodes ne nécessitent pas de maillage de géométries, mais se basent sur des sous-domaines de la géométrie. C'est le cas des méthodes de Monte Carlo. Ainsi, le format d'un maillage de référence, doit permettre de définir les maillages de chaque méthode de calcul.

Supposons qu'il existe un modèle de maillage de référence, compatible avec les différents maillages existants[7]. Pour mettre en œuvre ce type de schéma d'échange à l'aide d'un tel modèle, il est nécessaire qu'il existe un moyen technique permettant de traduire un format de maillage d'une simulation, au format de maillage de référence, et cela pour chaque simulation. Ensuite, une méthode d'intersection est mise en œuvre pour réaliser le maillage de référence. Cela demande, pour adapter une simulation au modèle d'irradiation des dispositifs, que les physiciens connaissent à la fois le format du maillage de leur outil de simulation et le format du maillage de référence. De plus la réalisation des moyens techniques de passage d'un format à l'autre peut contraindre à des développements informatiques lourds.

6.2.3 Schéma d'échange par interface

A contrario, ce schéma d'échange par *interface* permet de s'affranchir d'un maillage de référence. Il contraint chaque discipline à l'implémentation d'une seule méthode, ne supposant aucune connaissance d'un autre format de maillage.

Les codes de calcul utilisés par les physiciens, ont en chaque point de l'espace toute l'information concernant une quantité calculée (un résultat) sur un maillage spécifique. Il existe généralement des procédures, qui permettent d'accéder à ces résultats ponctuels, et certains codes de calcul permettent même de reconstruire les résultats sur des sous-ensembles de la géométrie (ou des maillages). Les physiciens disposent donc de moyens techniques suffisants, leur permettant de s'affranchir d'algorithmies géométriques, pour extraire des résultats de calcul sur un sous-domaine géométrique (par exemple un triangle, un segment, voire un maillage complet de la géométrie).

[7] Hormis les maillages présents en Neutronique avec la méthode des caractéristiques et les méthodes de Monté Carlo, le format MED œuvre en ce sens.

Nous proposons alors une unique méthode d'échange de données, nommée *getResults(element, identifiant d'un résultat, ...)*, qui permet de réaliser une requête pour obtenir la projection d'un champ de valeurs sur un élément simple (point, segment, triangle, quadrangle, tétraèdre, ...). La *projection* étant propre au champ de valeurs, à la discipline et à la méthode de calcul employée par la simulation, son développement est donc laissé à la charge des physiciens : chaque code de calcul (ou schéma de calcul) doit implémenter une méthode *getResults* capable de fournir, dans un format standardisé, l'information attendue par un quelconque autre code de calcul (ou schéma de calcul).

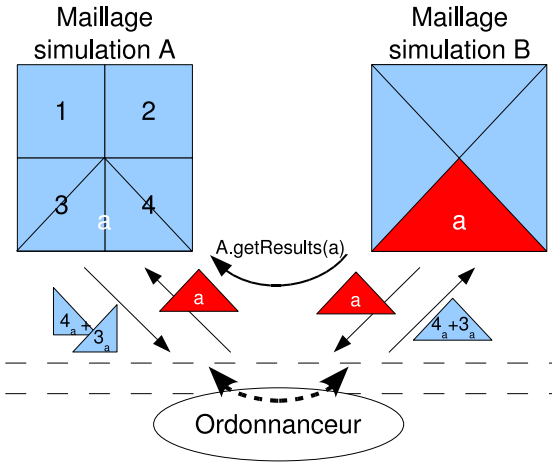


FIG. 6.6 – Principe d'échange de champs sans maillage de référence.

Le principe d'échange de données par interface est illustré par la figure 6.6. Considérons un enchaînement de deux simulations nommées A et B. Deux maillages différents, du même domaine, sont préalablement définis pour chacune de ces simulations. Supposons que pour réaliser un calcul, la simulation B doive initialiser un champ de valeurs connu de A. Alors, pour tous les éléments *simples* du maillage de B, est réalisée une requête à la simulation A, qui tient compte de la position et de la forme de chaque élément a du maillage B.

Par ailleurs, on constate que des algorithmes similaires auraient été mis en œuvre dans le cas de l'adoption d'une méthode d'échange par maillage de référence. En effet, la solution de maillage de référence consiste à projeter un maillage A sur un maillage B. Dans la pratique, le développement d'une procédure de projection suppose finalement de connaître pour chacune des mailles de B, une valeur équivalente portée par le maillage A.

Afin de garder le contrôle sur ces échanges de données, cette requête transite par un objet informatique qui a pour rôle d'ordonnancer les calculs et les échanges de données. Dans la pratique, la simulation B ne demande pas de résultats à la simulation A, mais à un ordonnanceur. Cet ordonnanceur a la connaissance que c'est la simulation A qui contient le résultat demandé. Ainsi, il est possible de contrôler divers cas de figure, dont en particulier, les cas où les résultats demandés n'ont pas encore été calculés. L'ordonnanceur a alors la capacité de synchroniser les calculs en mettant en état d'attente les simulations trop en avance sur le schéma de calcul en temps.

6.2.4 Discussion sur les schémas d'échanges

Alors que le schéma d'échange par maillage de référence consiste à définir, de manière explicite, un maillage résultant de l'intersection des maillages des simulations, le schéma d'échange par interface réalise de manière non-explicite une intersection entre deux maillages. L'avantage de ce dernier vient du fait que pour adapter une simulation à la plate-forme, il n'est pas nécessaire pour les physiciens d'avoir une connaissance d'un format de maillage de référence.

La tendance actuelle des développements des nouveaux codes de calcul, est de proposer le format MED comme standard pour les maillages d'entrées et de sorties des simulations, ainsi que pour les champs de valeurs. En adoptant un schéma d'échange par interface, les développements de la méthode *getResults* peuvent être distribués à l'ensemble des codes ayant suivi cette tendance et leur adaptation à la plate-forme d'irradiation des dispositifs expérimentaux sera quasi-immédiate.

Les principales caractéristiques d'un modèle multiphysique d'irradiation des dispositifs sont maintenant référencées. Le chapitre suivant détaille les développements mis en œuvre pour élaborer l'architecture informatique, intégrée dans un composant SALOME, de la plate-forme de modélisation multiphysique des dispositifs expérimentaux.

Chapitre 7

Mise en œuvre de l'architecture

Les contraintes de description géométrique présentées au chapitre précédent ont mis en évidence l'intérêt de l'utilisation d'un modèle orienté Features. C'est dans cette optique que le travail de thèse a été mené pour développer le modèle générique d'irradiation des dispositifs. Ce modèle est composé d'atomes de modélisation, que nous appelons Entités Technologiques. Les Entités Technologiques sont développées dans un composant (voir le paragraphe 4.3) dénommé TECHENTITY, dont l'objectif est de permettre la description d'une expérience d'irradiation dans l'environnement informatique de SALOME.

Ces développements sont inspirés des Objets Technologiques, mais diffèrent notamment de ceux-ci dans la manière de construire la géométrie. Le principe de cette modélisation est décrit en détail dans les quatre premiers paragraphes de ce chapitre. Les Entités Technologiques disposent d'objets permettant d'organiser le séquencement des simulations et également d'un mécanisme de gestion de versions, qui seront successivement décrits aux paragraphes 7.1.5 et 7.1.6. L'utilisation du composant TECHENTITY est présentée au paragraphe 7.2.

7.1 Les Entités Technologiques

La plate-forme de modélisation des dispositifs se place dans le cadre de l'IAO (simulations numériques) et également dans le cadre de la CAO (modélisation d'objets géométriques). Pour réaliser un modèle à l'interface CAO/IAO, nous proposons de développer le modèle générique des dispositifs d'irradiation nommé modèle d'*Entités Technologiques*.

Les entités technologiques sont développées dans un composant informatique, appelé TECHENTITY, qui permet de décrire une expérience d'irradiation à travers une interface graphique interactive ou bien sous forme de script en langage PYTHON [vR96]. Le composant TECHENTITY est inséré dans la plate-forme SALOME [Cas07b], ce qui permet de disposer d'un environnement composé de nombreux outils de CAO, de maillages, d'une interface graphique, d'une interface PYTHON, mais aussi de formats d'échanges stables et largement utilisés dans les domaines du calcul numérique et de la CAO (STEP, IGES, MED, UNV, ...).

Ces développements sont inspirés des Objets Technologiques dans la mesure où une Entité Technologique est une brique élémentaire d'un modèle décrivant un dispositif. Elle associe un matériau et une géométrie. Toutefois, il existe des différences de principes pour répondre aux spécifications des simulations multiphysiques des dispositifs expérimentaux.

7.1.1 Généralités

Une Entité Technologique est une brique élémentaire (atome de modélisation) du modèle de données. Elle est l'association d'une forme géométrique, d'une transformation géométrique et d'un matériau. Ce couple d'informations (géométrie, matériau) représente par exemple une pièce. Le matériau nous renseigne sur les propriétés physiques de cette pièce (telles que la conductivité thermique, la composition isotopique, le module d'Young...).

Une Entité Technologique peut être composée d'Entités Technologiques dites *filles* (attribut `children`), et peut avoir une relation avec une autre Entité Technologique dite *mère* (attribut `mother`). De cette manière, un dispositif expérimental est décrit par un arbre d'Entités Technologiques. Les simulations multiphysiques sont prise en compte au travers de la mise en relation d'une Entité Technologique avec une *séquence de calculs*. Cette *séquence de calculs* a alors pour rôle d'ordonnancer un ensemble de simulations relatives à cette Entité Technologique.

Afin de considérer de possibles modifications d'un modèle d'Entités Technologiques, un mécanisme de gestion de versions des modèles est mis en place. Celui-ci permet de modifier une partie ou l'ensemble du modèle tout en gardant un historique des modifications.

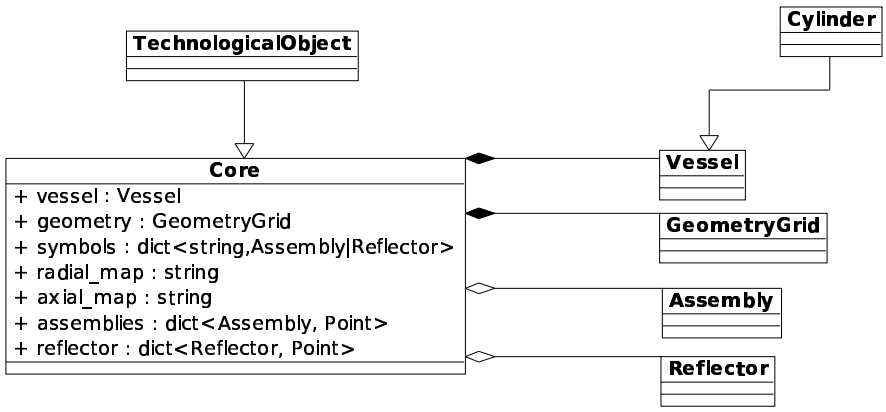


FIG. 7.1 – Diagramme UML d'un cœur (Objets Technologiques DESCARTES)

Les Entités Technologiques diffèrent des Objets Technologiques sur plusieurs points. D'une part, a contrario de la définition même des Objets Technologiques, la notion de résultats de calculs est présente dans les Entités Technologiques par la relation qui les lie à des séquences de calculs, présentées au paragraphe 7.1.5. D'autre part, les Objets Technologiques ont un modèle défini au préalable[1] et l'utilisateur se limite à la paramétrisation d'une instance de cet Objet Technologique. En reprenant les exemples présentés au paragraphe 4.5, l'utilisateur crée une instance de `Assembly` et paramètre

[1] Lors de la définition de la classe Assembly par exemple

les crayons combustibles (`FuelRod`). L'attribut `rods` (figure 4.3) de l'instance de la classe `Assembly` lui permet d'accéder aux instances qui représentent les crayons. Pour une instance de la classe `Core` (figure 7.1), qui représente le cœur d'un réacteur et ses assemblages, c'est l'attribut `assemblies` qui lui permet d'accéder aux instances représentant les assemblages.

Il est donc nécessaire pour l'utilisateur[2] de connaître le modèle de chaque instance qu'il manipule pour les paramétrer, tandis que les Entités Technologiques possèdent la même définition[3] quel que soit l'objet modélisé (que ce soit un crayon, une partie du réflecteur, un assemblage...). Elles supposent seulement certaines règles de création des objets auxquelles l'utilisateur devra se soumettre lors de la modélisation d'un dispositif.

7.1.2 Principes de modélisation par Entités Technologiques

Le principe de modélisation géométrique est décrit par la Figure 7.2 : à la géométrie d'une Entité Technologique sont soustraites[4] les géométries de ses Entités Technologiques filles.

Les objets géométriques construits par le composant GEOM, permettent d'effectuer des opérations géométriques et peuvent être traduits dans des formats de CAO[5]. Cependant, une fois construits, ces objets ne peuvent pas être modifiés. A contrario, la *géométrie locale* d'une Entité Technologique est un objet informatique permettant de décrire une géométrie à travers divers paramètres. Après la construction d'un objet, les modifications de ses paramètres influent directement sa représentation géométrique. Les opérations géométriques que peuvent réaliser ces objets sont limitées. Néanmoins, ces objets peuvent être traduits en objet GEOM[6].

L'application de ce concept est illustré dans l'exemple de la figure 7.2 : l'Entité Technologique ET1 a une *géométrie locale* qui est définie par l'utilisateur. Dans notre exemple, il s'agit d'un ovoïde percé d'un trou ovale. De même, des géométries locales sont définies pour les Entités Technologiques ET2 et ET3.
La contruction de la géométrie de ET1 est réalisée en lui supprimant les géométries de ET2 et ET3. Ces opérations sont effectuées sur des objets GEOM, qualifiés de *géométries calculées* (`ComputedGeometry`). La géométrie du dispositif complet est la composition de la géométrie résultante de cette opération avec les géométries de ET2, ET3.

Ainsi, pour construire le modèle d'un objet (objet modélisé), l'utilisateur définit un modèle plus simple à mettre en œuvre (modèle paramétré). A cet effet, il décrit dans un premier temps les géométries locales au moyen de formes élémentaires (ou plus complexes) et des matériaux. Les couples (matériau, forme géométrique) sont réalisés au moyen d'Entités Technologiques. Ensuite, il définit les liens d'affiliations (*children* et *mother*) entre les Entités Technologiques pour construire l'arbre.
Les géométries sont positionnées dans l'espace soit de manière absolue, soit de manière

[2] comme pour le code de calcul qui interprète ces instances
[3] i.e. les même attributs et les même méthodes
[4] soustraction selon un modèle CSG
[5] step, iges, brep
[6] D'un point de vue technique, cette traduction est réalisée par une méthode de mise à jour automatiquement appelée à chaque modification d'un paramètre.

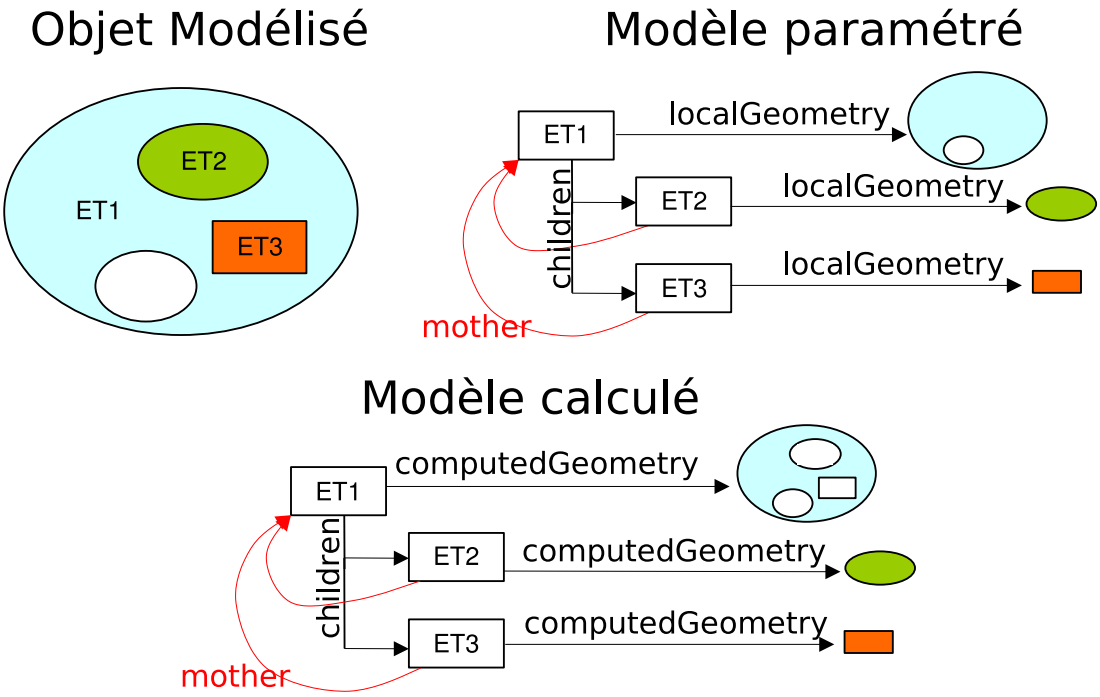


FIG. 7.2 – Principe de modélisation de la géométrie d'un dispositif par un arbre d'Entités Technologiques

relative à l'Entité Technologique mère en utilisant des transformations géométriques (voir 7.1.4).

La figure 7.3 illustre l'organisation des classes dans le composant TECHENTITY, permettant de réaliser le principe de modélisation de la figure 7.2. Les principales classes `TechnologicalEntity` , `Shape` et `Material` y sont décrites.

En comparant les attributs de la classe `TechnologicalEntity` avec ceux présentés dans la figure 7.2, on peut constater que bien que les attributs `localGeometry` et `computedGeometry` d'une même Entité Technologique représentent des objets géométriques, ceux-ci sont des concepts différents : l'un est un `Shape`, tandis que l'autre est un objet du composant GEOM. En effet, les objets référencés par `computedGeometry` sont des objets CAO du composant GEOM : il s'agit d'un modèle géométrique calculé (modèle calculé) à partir du modèle entré par l'utilisateur.

La classe `Material`, représentant les matériaux associés aux Entités Technologiques, est spécialisée en matériau solide (`Solid`), fluide (`Fluid`), et gazeux (`Gas`). Il est prévu la possibilité de considérer des compositions de matériaux. Cette fonctionnalité a pour but de permettre des simulations, par exemple en thermohydraulique, qui supposent des milieux diphasiques composés d'un même matériau se trouvant dans deux de ses états (par exemple liquide et gazeux). De même, des simulations numériques nécessitent des traitements numériques spécifiques aux matériaux combustibles (tels que l'autoprotection en neutronique, voir paragraphe 2.2). C'est pourquoi la classe `Solid` a été spécialisée

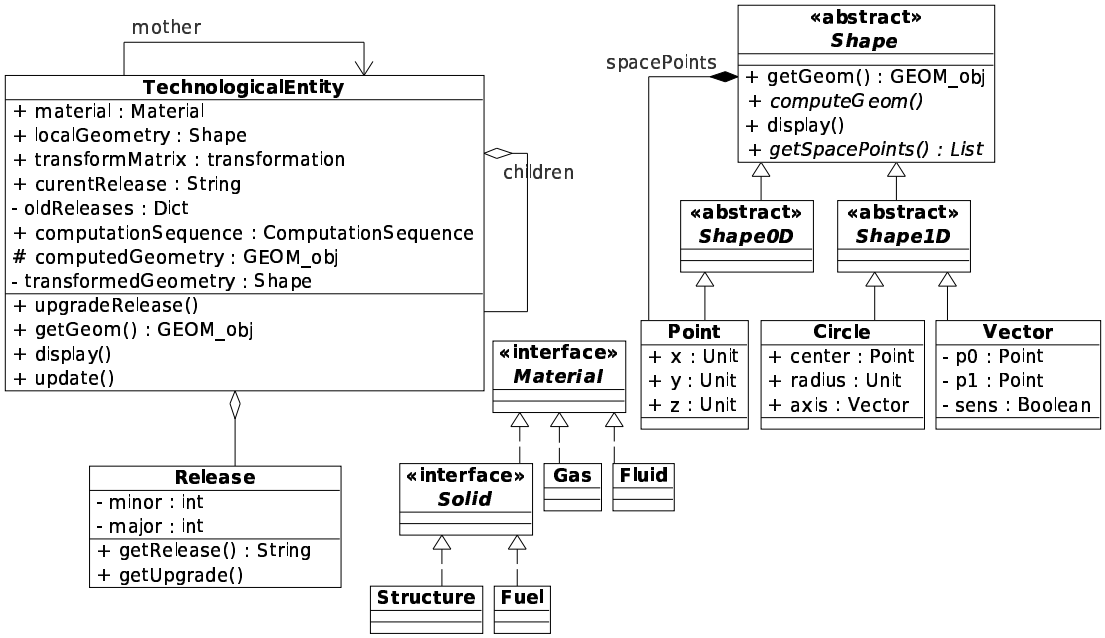


FIG. 7.3 – Diagramme de classe (non complet) des Entités Technologiques

en matériau de structure (`Structure`) et matériau combustible[7] (`Fuel`).

Durant la phase de développement du modèle d'un dispositif, chaque pièce est représentée par une Entité Technologique. Pour réaliser le modèle d'un dispositif expérimental avec un arbre d'Entités Technologiques, le dispositif est décomposé en pièces élémentaires, c'est-à-dire des pièces composées du même matériau. A chaque pièce élémentaire correspond une Entité Technologique, dont la géométrie locale et son matériau associé sont renseignés par l'utilisateur. Ensuite, seuls les liens descendants (ceux des Entités Technologiques filles) de chaque Entité Technologique sont définis. Ces opérations sont réalisées soit dans l'interface graphique de SALOME, soit en script en langage PYTHON. Une fois interprété, l'arbre d'Entités Technologiques est chargé en mémoire et sa représentation CAO est calculée[8]. Dès lors, une simple modification d'un attribut d'une Entité Technologique déclenche un appel à une méthode de mise à jour (méthode `update()`) qui peut être amenée à modifier l'ensemble de la géométrie du modèle.

7.1.3 Les formes géométriques : les spécialisations de la classe Shape

Un arbre d'Entités Technologiques décrit la géométrie d'un dispositif et dispose d'une méthode (`getGEOM()`) permettant d'interpréter cette description en un modèle CAO. Ce modèle est composé d'objets informatiques géométriques, décrivant des formes géométriques, qui héritent de la classe abstraite `Shape`. Les formes géométriques les plus habituelles (point, cercle, vecteur, plan, boite, cylindre, ...) ont été développées dans des classes qui implémentent `Shape`. Ces classes présentent donc une interface (un ensemble de méthodes et d'attributs) commune, qui permet de ne disposer que d'une seule méthode d'interprétation (`getGEOM()`) de l'arbre d'Entités Technologiques pour manipuler les objets géométriques. Ainsi, qu'il s'agisse d'un `Point(Shape0D)`, d'une forme

[7] Sous entendu combustible nucléaire

[8] Cette étape correspond à la création et au calcul des objets GEOM référencés par l'attribut computedGeometry

de dimension 1 (`Shape1D`), 2 (`Shape2D`), 3 (`Shape3D`) ou plus par la suite[9], la méthode d'interprétation des Entités Technologiques manipule les objets géométriques par les méthodes de `Shape`.

De la même manière, toutes les formes (`Shape`) peuvent-être affichées dans l'interface graphique de SALOME par la même méthode `display()` et les méthodes `getGeom()` permettent de récupérer des modèles équivalents CAO (objets du composant GEOM) afin de réaliser des traitements (coupes, exportations, ...).

Parmi les opérations effectuées par ces méthodes pour construire un modèle CAO, la méthode `computeGeom()` des objets `Shape` permet de construire un objet CAO sur lequel peuvent être effectuées des opérations géométriques. Dans le cas d'un cercle (`Circle`), cette méthode permet par exemple de contruire un objet CAO caractérisant un cercle à partir du centre, du rayon et d'un axe. En pratique, la méthode `computeGeom()` construit un objet géométrique à partir de commandes du module GEOM de SALOME.

Pour les placements dans l'espace et les transformations, l'attribut `spacePoints` contient une liste de `Point`. Il s'agit de points caractéristiques de la forme, tels que le centre d'un cercle ou les deux points qui caractérisent un segment. C'est à partir de ces points que les transformations géométriques sont réalisées.

Afin de concrétiser ces concepts, considérons le cas du développement d'une classe définissant un cercle : le cercle est un objet de dimension 1, de ce fait la classe `Circle` qui représente un cercle (figure 7.3), hérite de `Shape1D`. En ce qui concerne les attributs, un cercle est défini par un centre, un vecteur normal et un rayon. Nous définissons ces paramètres par les attributs `center`, `radius` et `axis`. La liste de l'attribut `spacePoints` doit être définie de manière à ce que les transformations appliquées sur ces points agissent sur le cercle. Elle contient dans ce cas une référence sur le centre du cercle et sur les points `p0` et `p1` de `Vector`.

En prenant le soin de développer la méthode `computeGeom()` de manière à construire la géométrie de l'objet à partir des points renseignés par `spacePoints`, une transformation des points de la liste `spacePoints` réalisera une transformation de l'ensemble du cercle. La méthode `computeGeom()` est implémentée dans la classe `Circle` ci-dessous :

```
1  class Circle(Shape1D):
2      __init__xattributes__ = [
3          XAttribute("radius", xtype=XFloat(open_min=0.0)),
4          XAttribute("center",xtype=XInstance(Point)),
5          XAttribute("axis",xtype=XInstance(Vector)),
6          ]
7      __object__xattributes__ = []
8      ...
9      ...
10     def computeGeom(self):
11         import batchmode_geompy
```

[9] Dans les développements actuels, les géométries sont supposées dans un espace à trois dimensions.

```
12          b_geompy = batchmode_geompy
13          radius = self.radius
14          center = self.center
15          axis = self.axis
16          self.geom_obj=b_geompy.MakeCircle(
17                  center.getGeom(),axis.getGeom(),radius)
18          return
19
20      def getSpacePoints(self):
21          return self.center.getSpacePoints()
22                  + self.axis.getSpacePoints()
23      ...
24      ...
```

C'est véritablement cette méthode qui permet de calculer le modèle CAO grâce au composant GEOM. Dans le cas de la définition de `Circle`, le module du composant GEOM est importé à la ligne 11 et le modèle CAO est créé aux lignes 16 et 17 par la méthode `MakeCircle` de GEOM. Cette commande demande en paramètres d'entrée deux objets de GEOM représentant le centre du cercle et l'axe normal du cercle ainsi que le rayon (réel). Les attributs `center` et `axis` sont de type `Shape`, et l'appel à la méthode `getGeom()` sur ces attributs permet de construire leurs modèles CAO qui sont transmis en argument de `MakeCircle`. Le modèle CAO construit, il est référencé par un attribut d'instance (privé) `geom_obj`. C'est cet attribut qui est retourné par la méthode `getGeom()`.

Toutefois, il est à considérer que cet exemple est un cas simplifié : on pourra constater qu'une transformation telle qu'une rotation ou une translation, préalablement appliquée aux points de `spacePoints`, s'appliquera sur l'ensemble du cercle. Néanmoins, une transformation qui engendrerait une dilatation ne serait pas appliquée convenablement. En effet, une dilatation du cercle implique une modification du rayon et celui-ci n'est pas défini en fonction de points présents dans `spacePoints`. Une implémentation permettant de prendre en compte cette dernière remarque est détaillée en annexe F.

Ces objets sont paramétriques, il est nécessaire de contrôler les accès en lecture et en écriture sur les attributs de ces objets. Du fait qu'il s'agisse de `XObjet`, les accesseurs des attributs sont transparents pour l'utilisateur et préimplémentés pour le développeur. Ainsi, dans le cas de l'attribut `radius`, toute attribution fait appel à la méthode `setRadius` qui peut-être surchargée. La lecture de ce même attribut appelle la méthode `getRadius`. Les appels aux méthodes de mise à jour du modèle, par exemple l'appel à la méthode `computeGeom()`, sont implémentés dans ces accesseurs.

7.1.4 Le positionnement et les transformations géométriques : Les matrices de transformation

Nous traitons ici le cas de positionnement relatif des géométries. Pour cela, une transformation doit être associée à une Entité Technologique.

En pratique, une transformation peut-être décrite via une matrice τ_{ra}. Le point P' image

de P par la transformation décrite par τ_{ra} est défini comme suit :

$$P' = [\tau_{ra}] \cdot P$$

Généralement[10], ces opérations sont réalisées en coordonnées homogènes, c'est-à-dire dans un espace de dimension $n+1$. Ainsi, un point de coordonnées (x, y, z) correspond à $(x, y, z, 1)$. Dans le cas de géométries placées dans un espace de dimension 3, τ_{ra} est une matrice 4×4. Les matrices de transformation usuelles sont définies en annexe E.

Lors de la construction de la géométrie d'une Entité Technologique, sa géométrie locale et tous les objets géométriques dont elle est dépendante, sont dupliqués puis transformés. Le mécanisme de transformation proposé dans le développement des Entités Technologiques applique ces transformations à chaque point de la liste `spacePoints` d'un objet `Shape`.

Considérons l'exemple d'un simple cylindre devant subir une transformation géométrique. Cet objet est modélisé par une Entité Technologique dont la géométrie locale est un objet `Cylinder`. Lors de la construction de la géométrie, l'instance de l'attribut `localGeometry` est dupliquée, dans ce cas il s'agit de l'objet `Cylinder`. Pour construire une instance de `Cylinder`, il est nécessaire de renseigner les attributs `center`, `axis`, `radius` et `height`. Les deux premiers attributs étant des instances de `Shape`, celles-ci sont à leur tour dupliquées. La liste `spacePoints` du `Cylinder` retourne les points correspondant à la position du cylindre dans l'espace, ainsi que les deux points de l'axe du cylindre. Ces points sont alors modifiés par leurs images issues de la transformation géométrique. Puisque ces objets sont paramétriques, ces modifications affectent directement la représentation de la géométrie. En effet, les modifications de ces objets sont réalisées à travers des accesseurs transparents pour l'utilisateur. Ces accesseurs réalisent, si nécessaire, des appels aux méthodes de mise à jour du modèle de données.

7.1.5 Les séquences de calculs

Organisation

Le terme *simulation* est employé ici pour désigner un schéma de calcul (ou jeu de données) dédié à la simulation d'un objet particulier dans un domaine de physique précis. Les codes de calcul sont considérés comme des paquetages contenant des fonctions de calculs dédiés à une discipline, utilisables par une interface ou un langage spécifique. Un schéma de calcul est en quelque sorte un programme informatique qui utilise un paquetage précis : le code de calcul. Certains schémas de calcul sont développés de manière à permettre la paramétrisation de données d'entrée et de sortie.
Une séquence de calcul est un enchaînement de schémas de calcul pouvant être amenés à échanger des données.

Les développements de la thèse ont été dirigés pour s'appliquer plus particulièrement aux schémas de calcul qu'aux codes de calcul. Il s'agit d'outils d'aide au couplage explicite (voir paragraphe 2.4) de schémas de calcul. Ainsi les adaptations des simulations

[10] Afin de pouvoir travailler sur des points placés à l'infini et de ne pas effectuer des traitements particuliers pour le parallélisme.

à notre architecture sont à la charge des physiciens qui développent les schémas de calcul.

Afin de réaliser un enchaînement automatisé de simulations, nous proposons de mettre en place des méthodes standard pour les échanges de données et les traitements. Parmi les traitements, nous entendons d'une part les mises en forme des résultats de calcul, d'autre part l'ensemble des méthodes qui permettent le contrôle des simulations par un ordonnanceur. Il a été choisi d'utiliser comme méthodes standard, les méthodes identifiées par les interfaces informatiques proposées par Fabien Perdu dans [Per06] (figure 7.4). L'ordonnanceur est représenté par une classe `ComputationSequence` (illustrée par la figure 7.5).

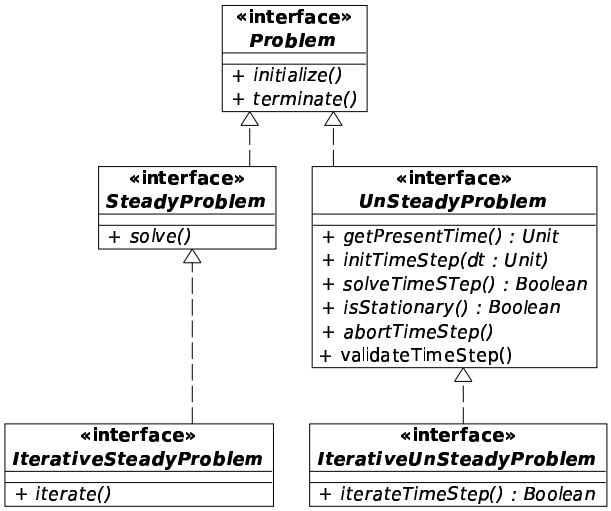


FIG. 7.4 – Diagramme de classe de Problem (inspiré de [Per06])

Pour adapter une simulation à notre architecture, de sorte qu'elle puisse être pilotée par un objet `ComputationSequence` et ainsi échanger des grandeurs avec d'autres simulations, nous proposons qu'elle soit encapsulée dans une des interfaces de `Problem` (figure 7.4). Cette interface et ses spécialisations ont pour objectif de définir des méthodes de contrôle des simulations. Elles ont été définies pour faciliter le couplage (implicite) de codes de calcul. Dans les travaux de thèse, nous développons des objets de même définition, que nous utilisons pour coupler (couplage explicite) des schémas de calcul. Ainsi, toutes les simulations d'un objet `ComputationSequence` implémentent les méthodes de `Problem`.

Cette première interface n'a que deux fonctionnalités très générales permettant pour l'une d'initialiser (`initialize`) et pour l'autre de terminer (`terminate`) un calcul. Les spécialisations de `Problem` proposent des méthodes dédiées à la nature de la simulation et à ce que les physiciens souhaitent contrôler. Ces méthodes peuvent aller d'une résolution complète d'un problème stationnaire (`SteadyProblem`) au contrôle des boucles internes lors de la résolution du problème (`IterativeSteadyProblem`), ou bien au contrôle des pas de temps d'une simulation (`UnSteadyProblem`). A ce jour, quatre interfaces sont proposées par [Per06] :

– l'interface `SteadyProblem` qui traite des problèmes stationnaires. Elle ajoute une

méthode de résolution du problème (`solve`) ;
- l'interface `UnSteadyProblem` qui traite des problèmes non stationnaires, c'est-à-dire dépendant du temps. Cette interface permet de contrôler la résolution du problème par pas de temps.
- l'interface `IterativeSteadyProblem` qui traite des problèmes stationnaires en ajoutant un contrôle sur les itérations des méthodes de résolution.
- l'interface `IterativeUnSteadyProblem` qui traite des problèmes non stationnaires et ajoute pour chaque pas de temps un contrôle sur les itérations des méthodes de résolution.

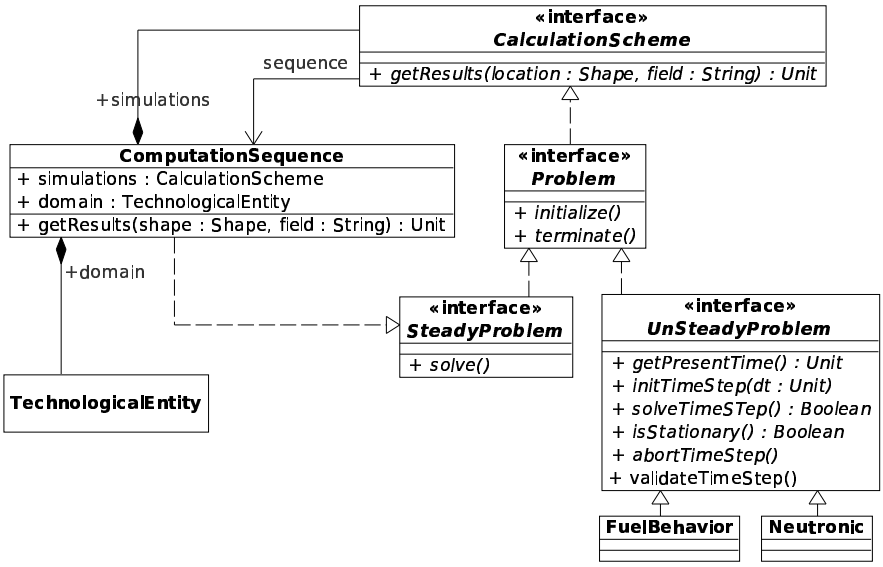


FIG. 7.5 – Diagramme de classe de ComputationSequence et CalculationScheme

En ce qui concerne l'échange des résultats et le traitement des résultats de calculs, cela est assuré par l'interface `CalculationScheme` dont l'interface `Problem` hérite des fonctionnalités[11]. En analysant la figure 7.5, on en déduit qu'une simulation (un schéma de calcul encapsulé dans une des interfaces `Problem`) *est un* `CalculationScheme`. Elle accède donc à la séquence de calcul par l'association `sequence` (de `CalculationScheme`) et peut aussi accéder[12] à l'arbre d'Entités Technologiques à travers de la composition `domain`.

La méthode `getResults`, proposée au paragraphe 6.2.3, est quant à elle plus spécialisée dans l'échange de champs de valeurs entre simulations. Elle permet de retourner un vecteur de données correspondant à un champ de valeurs, calculé en un lieu et sur un domaine[13] précis. L'intérêt de cette méthode intervient dans l'exemple suivant : considérons deux simulations d'un même objet, traitant de problèmes de physique différents. De plus, ces deux simulations réalisent leurs calculs à partir de maillages différents du domaine d'étude. Une séquence de calcul ordonnance ces simulations en les exécutant séquentiellement. Ainsi, la première des deux simulations est lancée et les résultats de calcul sont sauvegardés. Ensuite, la seconde simulation nécessite une partie des résultats

[11] Il est surtout question de la méthode getResults.

[12] En lecture et en écriture, ce qui implique que le modèle d'arbre d'Entités Technologiques peut être modifié par des résultats de calcul.

[13] C'est à dire une forme géométrique telle qu'un triangle, un quadrangle, un tétraèdre...

de calcul de la première simulation. Lors de l'initialisation de la seconde simulation, un appel à la méthode `getResults` *est demandé*[14] pour un champ particulier et sur une partie du domaine, par exemple un élément de type triangle appartenant au maillage de calcul. La première simulation, traite alors ses résultats de calcul de manière à fournir un résultat sur ce triangle. Ainsi, il n'aura pas été nécessaire de convertir la totalité des résultats de la première simulation sur le maillage de la seconde simulation.

Il est indéniable que la manière dont la méthode `getResults` traitera les résultats de calcul d'une discipline va dépendre de la simulation abordée. Ainsi, l'implémentation de la méthode `getResults` est à la charge des physiciens. En effet, des interpolations des résultats sont nécessaires, celles-ci peuvent supposer des hypothèses sur l'interprétation physique souvent différentes d'un champ de résultats à l'autre. Prenons comme exemple, la cas suivant :

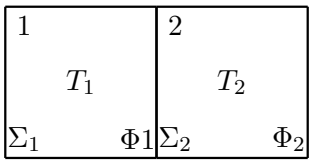


FIG. 7.6 – Exemple de résultats d'un calcul neutronique sur un domaine composé de deux milieux différents.

En neutronique, on souhaite extraire les champs de résultats $\Phi_{1,2}$, représentant le flux de neutrons, et $T_{1,2}$ représentant un taux de réaction sur un domaine. Le maillage de calcul de ce domaine est composé de deux régions, indicées 1 et 2 (voir figure 7.6), dont chacune a un comportement neutronique différent, caractérisées par des sections efficaces Σ_1 et Σ_2 différentes. Considérons que les paramètres de neutronique ont précédemment été calculés, il existe alors les résultats de calculs T_1, T_2, Φ_1 et Φ_2.
Pour le champ correspondant au taux de réaction, la simple relation suivante permet de retourner le résultat demandé :

$$T_{1,2} = T_1 + T_2$$

En ce qui concerne le champ correspondant à un flux, la relation est différente :

$$\Phi_{1,2} = \frac{\Sigma_1\Phi_1 + \Sigma_2\Phi_2}{\Sigma_{1,2}} \left(= \frac{T_{1,2}}{\Sigma_{1,2}} \right)$$

En considérant au préalable que $\Sigma_{1,2}$ a été évalué. On s'aperçoit donc que deux champs de résultats d'un même domaine, ne sont pas traités de la même manière.

Toutefois, des outils génériques d'interpolation [Gra04] de résultats enregistrés au format MED sont en développement, et il est envisageable de les utiliser dans la mesure où les résultats de calcul sont exportés dans ce format.

[14] Afin d'assurer le contrôle des simulations, l'appel à la méthode getResults de la première simulation est réalisé par la séquence de calcul.

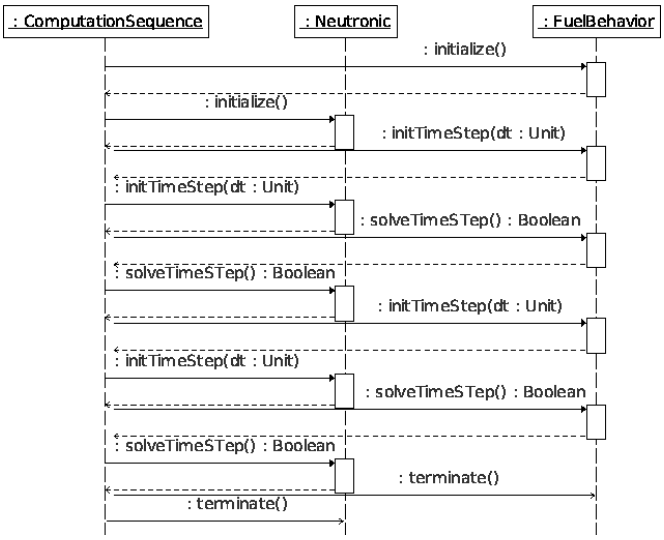


FIG. 7.7 – Diagramme de séquence d'une séquence de simulations (deux pas de temps).

Séquencement

Comme le montre la figure 7.5, la classe `ComputationSequence` implémente[15] l'interface `SteadyProblem` . Cette dernière présente une méthode `solve`. Ainsi pour lancer une séquence de simulations, il suffit d'exécuter cette méthode.

Considérons une séquence de calculs composée de deux simulations, l'une traitant de neutronique (`Neutronic`), l'autre traitant de comportement combustible (`FuelBehavior`). Ces simulations sont encapsulées dans des interfaces `UnSteadyProblem` (ces interfaces sont présentées sur la figure 7.5) et un utilisateur crée un objet `ComputationSequence` composé de ces deux simulations. Par ailleurs, cet objet est associé à un arbre d'Entités Technologiques qui décrit le domaine d'étude. La figure 7.7 présente une simple séquence d'appel des méthodes des simulations, réalisée par la méthode `solve` de la séquence de calcul. En premier lieu, les méthodes d'initialisation de chaque simulation sont appelées. Cette étape consiste par exemple, à configurer les calculs, les données d'entrées et à charger les codes de calcul. L'exécution des programmes de simulation est alors mise en attente avant les calculs par pas de temps. Ensuite, les méthodes `initTimeStep` et `solveTimeStep` sont appelées. La première de ces méthodes permet d'initialiser les données d'entrées spécifiques au pas de temps actuel. La seconde méthode résout le problème sur le pas de temps considéré. Pour terminer cette boucle de pas de temps, la méthode `terminate` est appelée.

La figure 7.8 présente la même séquence de calcul avec cette fois des exemples d'échanges de données, soit avec l'arbre d'Entités Technologiques, soit par le biais de la méthode `getResults`. Dans l'exemple, lors de l'initialisation du comportement combustible, le rayon d'un crayon est lu dans le modèle d'Entités Technologiques. Sur le même principe, lors de l'initialisation du calcul neutronique, une représentation CAO de l'arbre d'Entités Technologiques est récupérée par la simulation. Il s'agit dans ce cas d'un objet du composant GEOM, pouvant être exporté vers divers formats.

[15] C'est-à-dire que la classe ComputationSequence donne un sens qui lui est propre à chaque méthode définie par l'interface.

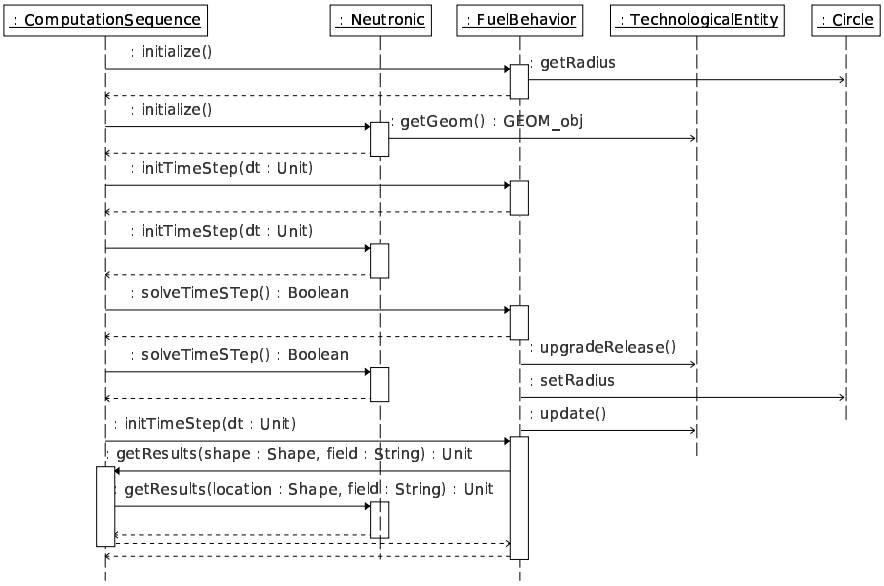


FIG. 7.8 – Diagramme de séquence d'une séquence de simulations avec des échanges de données.

Ensuite, les méthodes `initTimeStep` et `solveTimeStep` sont appelées. Il est alors possible que les calculs du comportement combustible prédisent une modification du rayon d'un crayon combustible. Durant une prochaine étape de validation[16], ces modifications sont prises en compte dans l'arbre d'Entités Technologiques. Ainsi une nouvelle version du modèle d'Entités Technologiques est créée par l'appel de la méthode `upgradeRelease()`.

La méthode `getResults` est appelée sur la figure 7.8 lors de l'initialisation du second pas de temps. Supposons que la puissance délivrée par le combustible (essentiellement des réactions de fission) soit une donnée d'entrée, dépendante du temps, de la simulation traitant du comportement combustible. Cette puissance est calculée par la neutronique. L'appel de la méthode `getResults` de la séquence de calcul est alors réalisé par la simulation (`FuelBehavior`), en précisant le champ (dans ce cas il s'agit de la puissance) et le lieu. La séquence de calcul, instance qui ordonnance les simulations, se dirige vers la simulation neutronique pour renseigner des résultats.

Ainsi, la simulation du comportement combustible n'a pas *connaissance* de la simulation de neutronique. Celle-ci peut être remplacée par tout autre modèle, et il en est de même pour le comportemment combustible. Une application directe pourrait être de comparer les calculs de deux modélisations différentes[17] du même phénomène physique. Il est aussi possible de substituer un schéma de calcul par une base de données expérimentale, encapsulée dans une interface `Problem`.
De plus, une séquence de calcul, instance de `ComputationSequence`, implémente une des interfaces `Problem`. Elle peut alors être utilisée telle une simulation et ordonnancée par une autre séquence de calcul.

[16] Par souci de lisibilité, l'appel aux méthodes validateTimeStep n'est pas précisé. Ces appels sont réalisés par la séquence de calcul pour chaque simulation.
[17] Par exemple basée sur des codes de calcul différents

7.1.6 La gestion de versions des arbres d'Entités Technologiques

Afin de proposer un moyen permettant d'assurer en partie la gestion du cycle de vie du dispositif, et de permettre de sauvegarder des modifications du modèle lors des simulations, un mécanisme de gestion de version de l'arbre d'Entités Technologiques est mis en place. Le principe s'apparente à des calques d'études qui permettraient de modifier la totalité ou bien une partie de l'arbre. La possibilité d'ajouter ou de retirer des objets a été prévue.

Ce mécanisme permet de sauvegarder les attributs `children`, `mother`, `localGeometry`, `Material` et `transformMatrix` des Entités Technologiques modifiées. On suppose a priori que le modèle d'un dispositif est décrit par un arbre d'Entités Technologiques. L'appel de la méthode `upgradeRelease()`[18] permet d'activer le mécanisme en associant à l'arbre courant, une version initiale 0.0 et en passant la version actuelle en 0.1 (attribut `curentVersion`). Dès lors, toutes les modifications de l'arbre par rapport à la version initiale sont sauvegardées.

D'une part, la sauvegarde des attributs impactant la géométrie *locale* d'un dispositif (`localGeometry`et `transformMatrix`) est assurée, et d'autre part la hiérarchie des Entités Technologiques dans l'arbre peut se trouver fortement modifiée d'une version à l'autre.

Les principaux concepts proposés par les Entités Technologiques ont été traités, le paragraphe suivant illustre l'utilisation des fonctionnalités de l'outil développé. Ces développements sont intégrés dans un composant SALOME que nous avons nommé TECHENTITY.

7.2 Le composant TECHENTITY

Le composant TECHENTITY regroupe un ensemble de fonctionnalités permettant de construire un modèle d'Entités Technologiques, et de réaliser un séquencement de calculs. Ce paragraphe présente différents aspects de son utilisation, dont ses fonctionnalités géométriques, la méthodologie de construction d'un arbre d'Entités Technologiques et de mise en œuvre d'une séquence de calculs.

7.2.1 Les fonctionnalités Géométriques

Les menus permettant d'accéder à la construction des objets géométriques sont présentés en annexe H.

Les objets géométriques sont groupés selon quatre sous-menus correspondant à des entités géométriques de dimension zéro à trois. Des objets géométriques 3D sont préalablement définis, ainsi des boites (`Box`), cylindres, sphères et tubes sont accessibles directement par le menu Geometry (3D). Pour des objets géométriques plus complexes,

[18] upgradeRelease() est une méthode de TechnologicalEntity (figure 7.3), cet appel peut être réalisé par l'interface graphique ou en PYTHON.

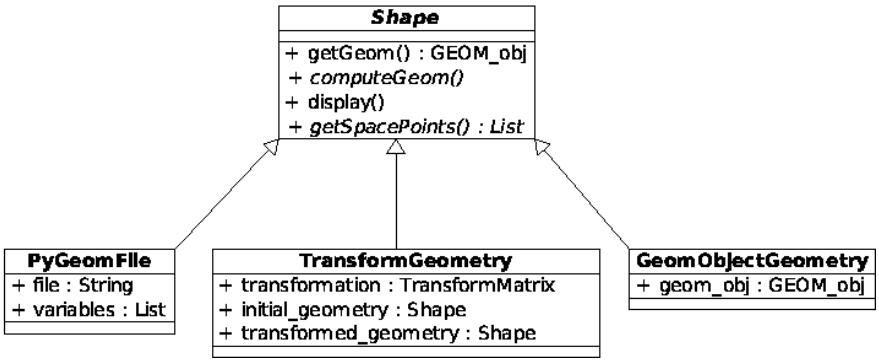


FIG. 7.9 – Digaramme de classe de TransformedGeometry, PyGeomFile et GeomObjectGeometry.

l'utilisateur peut construire des solides à partir d'objets géométriques de dimension inférieure comme il est décrit dans l'annexe I.

Un cinquième sous-menu, nommé *Spécial*, permet d'accéder à trois classes d'objets géométriques particuliers : `PyGeomFile`, `TransformGeometry`, `GeomObjectGeometry`. Ces classes d'objets ne sont pas spécialisées à une dimension : elles dérivent directement de `Shape` (voir figure 7.3 et figure 7.9).

La première d'entre-elles, `PyGeomFile`, permet d'encapsuler dans un objet `Shape`, un script PYTHON déjà existant, qui construirait un objet géométrique avec le composant GEOM. Cette encapsulation ne nécessite que de légères adaptations décrites dans l'annexe J.
La classe `TransformGeometry` associe un objet `Shape` (`initial_geometry`) à une matrice de transformation (objet de la classe `TransformMatrix` accessible par le menu *Opérations*). Ainsi, l'attribut nommé `transformed_geometry` (figure 7.9) référence une copie de `initial_geometry`, à laquelle à chacun des points de `spacePoints` sont attribués leurs images par la matrice de transformation géométrique. Lors de la copie d'un objet `Shape`, seuls les attributs de construction (i.e. ceux présents dans la liste `__init__xatttributes__`) sont copiés : il s'agit d'une copie profonde[19].
Enfin, la classe `GeomObjectGeometry` permet simplement de référencer un objet géométrique du composant GEOM de manière à pouvoir l'utiliser en tant qu'objet `Shape`.

Il est possible d'enrichir les objets géométriques, et d'ajouter des opérations ensemblistes (union, différence, intersection, ...). Dans l'absolu, l'ensemble des fonctionnalités du composant GEOM peut être encapsulé dans des objets équivalents du composant TECHENTITY.

7.2.2 La construction d'un arbre d'Entités Technologiques

La construction d'un arbre d'Entités Technologiques est réalisée en trois étapes. La première de ces étapes consiste à définir l'ensemble des objets géométriques et des matériaux nécessaires à la construction du modèle. La seconde étape est de réaliser les

[19] S'agissant d'une copie profonde, si les objets devant être copiés sont des *Shape*, alors une copie similaire est appliquée à ces objets.

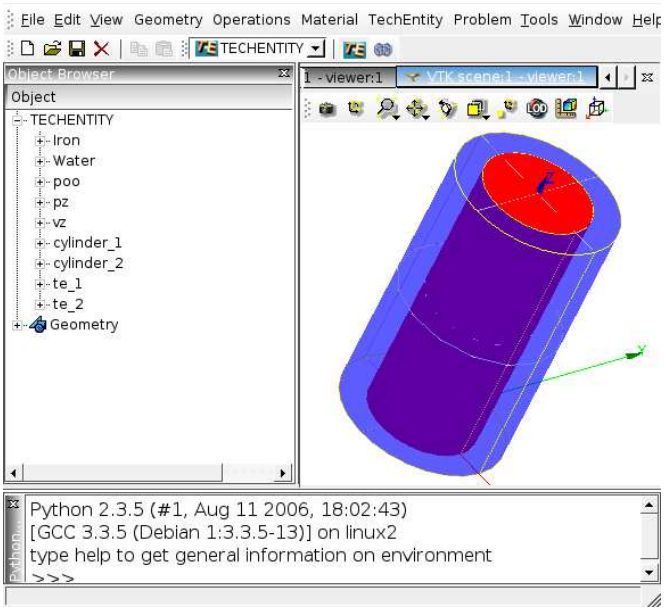


FIG. 7.10 – Modélisation par Entités Technologiques d'un tube en Fer contenant de l'eau et visualisation dans SALOME.

associations géométrie/matériau en construisant des instances de `TechnologyEntity`. La troisième et dernière étape permet de réaliser les liens mère/enfants (affiliation) des Entités Technologiques.

```
1 s1 = Schema1()
2 s2 = Schema2()
3 s3 = Schema3()
4 cS = ComputationSequence(simulations = [s1,s2,s3])
```

Par exemple, la construction du modèle présenté par la figure 7.10 peut être le résultat du script suivant (les importations de modules PYTHON ne sont pas précisées) :

```
# Etape 1 : Construction des objets geometriques
#                 et des Materiaux
Iron   = Structure()
Water  = Fluid()
poo    = Point(x=0.0,y=0.0,z=0.0)
pz     = Point(x=0.0,y=0.0,z=1.0)
vz     = Vector(p0=poo,p1=pz)
cylinder_1 = Cylinder(poo,vz,radius=3.0,height=10.0)
cylinder_2 = Cylinder(poo,vz,radius=2.0,height=10.0)

# Etape 2: Association des objets géométriques
#                 et des Materiaux

te_1 = TechnologyEntity(material=Iron,
                        localGeometry=cylinder_1)
te_2 = TechnologyEntity(material=Water,
                        localGeometry=cylinder_2)

# Etape 3 : Affiliation
```

```
te_1.children = [te_2]
```

Suite à ces développements, un exercice de modélisation du dispositif TANOXOS permet de valider la faisabilité du concept de modélisation proposé par la thèse. Le jeu de données (en Python) de cet exercice, construit un modèle d'Entités Technologiques décrivant le dispositif expérimental TANOXOS. Il est disponible en annexe K.

7.2.3 La mise en œuvre d'une séquence de calculs

Construction

Supposons qu'il existe des schémas de calculs, encapsulés dans des classes `Schema1`, `Schema2`, `Schema3` qui implémentent une des interfaces `Problem`. Dans ce cas, la mise en œuvre d'une séquence de calcul (`cS`) permettant l'enchaînement (voire le couplage) des schémas est réalisée par un objet `ComputationSequence` de la manière suivante :

```
s1 = Schema1()
s2 = Schema2()
s3 = Schema3()
cS = ComputationSequence(simulations = [s1,s2,s3])
```

Une Entité Technologique peut être associée à la séquence de calcul (référencée par l'attribut `domain` d'un objet `CompuationSequence`). Cette Entité Technologique est alors commune à chacun des schémas de calculs que compose la séquence de calcul : il est possible à partir de ces schémas d'accéder à cette Entité Technologique et de parcourir l'arbre auquel elle appartient.

Exécution

Techniquement, la classe `ComputationSequence` hérite de `SteadyProblem` (voir figure 7.3), elle dispose donc d'une méthode `solve()`, mais aussi des méthodes `initialize()` et `terminate()` de l'interface `Problem`. Ce sont ces méthodes qui permettent d'exécuter la séquence de calcul soit par le biais de l'interface graphique de SALOME (action bouton droit, voir figure 7.11), soit par les commandes suivantes :

```
cs.initialize()
cs.solve()
cs.terminate()
```

On constate qu'une séquence de calcul est contrôlable de la même manière qu'un Schéma de calcul : les séquences de calcul présentent la même interface que les schémas de calcul.

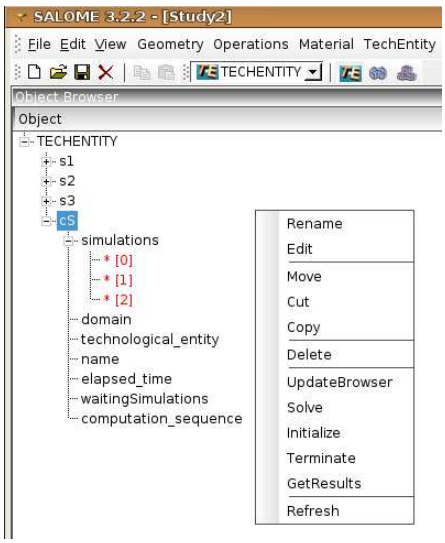


FIG. 7.11 – Utilisation d'une séquence de calcul au travers de l'interface graphique de SALOME.

Synthèse de la deuxième partie

Cette seconde partie a présenté les réflexions et la mise en œuvre de la plate-forme de modélisation des dispositifs expérimentaux pour des couplages multiphysiques.

Pour développer cette plate-forme nous avons cherché un modèle multiphysique de données capable de décrire un dispositif expérimental. A cette occasion, nous avons pu évaluer au chapitre 5, par un exercice d'application à un dispositif CHOUCA, le modèle de données composé d'Objets Technologiques, proposé par le groupe de travail PAL/DISCIPLINE. Ce modèle multiphysique de données, paramétrique et intégré dans SALOME, est spécialisé dans la modélisation de réacteurs électrogènes. Il est cependant moins bien adapté à la modélisation de dispositifs expérimentaux quelconques, car cela supposerait des développements informatiques à chaque modélisation d'un nouveau dispositif.

Suite à cette évaluation, il a été choisi de définir et de développer un modèle de données multiphysique qui s'inspire de certains concepts et solutions techniques adoptés par les Objets Technologiques. Ce nouveau modèle de données est intégré dans SALOME. Il est développé avec les outils XDATA proposés par le CEA pour faciliter le développement de composants SALOME.

Dans la chapitre 6 nous avons spécifié les besoins nécessaires à une modélisation pluridisciplinaire[20] de dispositifs expérimentaux. Ces spécificités nous ont amené à une discussion dans laquelle les choix techniques en termes de modélisation géométrique, d'échanges et de partage de données, ont été émis.

Dans le chapitre 7, les développements réalisés sont présentés. Le modèle de données multiphysique qui est proposé est paramétrique. Il permet, au moyen d'atomes de modélisation que nous nommons Entités Technologiques, de définir la géométrie d'un dispositif expérimental à partir de fonctionnalités dont dispose SALOME, et d'associer des informations concernant les matériaux des objets. Un dispositif expérimental est alors modélisé par un arbre d'Entités Technologiques.

A ceci s'ajoute une architecture permettant de standardiser les échanges de données entre les schémas de calculs. Cette standardisation est traduite par une encapsulation des schémas de calculs dans des interfaces informatiques (les interfaces `Problem`) et par une méthode commune aux schémas de calculs, qui renvoie un résultat localisé précisé-

[20] S'agissant de modélisations multiphysiques et géométriques, nous qualifions les modélisations de *pluridisciplinaires*.

ment dans le domaine géométrique (la méthode `getResults`).

Les développements intégrés dans un composant SALOME nommé TECHENTITY, ont été réalisés de manière à simplifier, pour les physiciens, les aspects informatiques de la modélisation pluridisciplinaire. Le paragraphe 7.2 présente quelques détails d'utilisation de ce composant.

Dans la troisième partie de ce document, nous verrons un cas d'application de ces nouveaux outils, sur une modélisation multiphysique du dispositif expérimental TANOXOS du réacteur OSIRIS. Cette étude permettra d'évaluer les apports des Entités Technologiques aux métiers des physiciens.

Troisième partie

Application à l'étude de dispositifs expérimentaux

Pour le développement et la validation des premiers résultats de la plate-forme de simulation des dispositifs expérimentaux, il a été décidé d'étudier le dispositif TANOXOS du réacteur OSIRIS. Ce dispositif, bien instrumenté, permet de balayer un éventail large de paramètres utilisables pour une éventuelle qualification des résultats des calculs qui poursuivrait le travail de thèse.

L'objectif de cette troisième partie est donc de mettre en application les outils précédemment développés, à partir d'un cas d'application concret sur un dispositif existant. Aussi a-t-il fallu s'impliquer dans les métiers des physiciens en développant et adaptant des modèles physiques appropriés.

Le premier chapitre de cette partie présente le dispositif expérimental TANOXOS et les modélisations physiques actuellement réalisées en routine à OSIRIS. Nous y présentons ensuite le modèle multiphysique proposé pendant la thèse.

Le dispositif TANOXOS est situé en périphérie du cœur, ceci implique que des difficultés de modélisation du flux neutronique en réflecteur, à cause de la forte anisotropie des chocs des neutrons sur l'hydrogène de l'eau, peuvent apparaître. Le travail de thèse présente donc également l'occasion de mettre au point un schéma de calcul neutronique innovant pour ce type de dispositif, fondé sur la méthode des caractéristiques. Cette étude est détaillée au chapitre 9.

Dans le chapitre 10, on considère alors que toutes les disciplines sont correctement encapsulées, et il est mis en œuvre un calcul multiphysique du dispositif TANOXOS à travers la plate-forme de modélisation des dispositifs expérimentaux.

C'est ici qu'intervient l'aspect pluridisciplinaire de la thèse : en plus de la définition et du développement de l'architecture de la plate-forme, des développements et des adaptations des modèles physiques appropriés ont été réalisés. La thèse se déroulant dans un laboratoire de neutronique, cette discipline sera abordée de manière plus approfondie dans cette partie avec des résultats complémentaires en annexe L.

Chapitre 8

Objectifs

8.1 Présentation du dispositif TANOXOS

Le dispositif TANOXOS (figure 8.1) est développé pour réaliser des irradiations qui ont pour objectif l'étude et la mise au point de nouveaux combustibles nucléaires à microstructures et compositions optimisées ([DCD+98]) dans le but de limiter le relâchement des gaz de fission.

Le dispositif est conçu pour recevoir des crayons combustibles de petite taille. Il peut accueillir différents types de crayons : UO2, MOX, CERMET, CERCER. La température à cœur des combustibles est représentative de celles que verraient ces combustibles en réacteur de puissance. Les irradiations se déroulent sur de longues durées, pouvant varier de 2 à 5 ans, et constituent une étape de vérification importante du comportement des combustibles à des taux de combustion élevés.

Le dispositif est situé en périphérie du cœur, l'évacuation de la puissance dégagée par les crayons irradiés étant assurée par circulation de l'eau de la piscine. Chaque charge expérimentale est constituée d'un crayon combustible formé d'une vingtaine de pastilles placées dans une gaine en inox pressurisée, elle même placée dans une surgaine en aluminium. L'instrumentation du dispositif est constituée de sept collectrons Rhodium dans le fourreau de refroidissement du dispositif (mesure de flux) ainsi que d'un thermocouple et d'intégrateurs de dose dans chaque surgaine et d'un thermocouple à cœur dans chaque charge expérimentale.

Les paramètres importants pour le bon déroulement d'une irradiation sont le respect de la consigne de température sur le crayon le plus chaud, et l'homogénéisation des taux de combustion des différents crayons pour réaliser la meilleure discrimination possible en fin d'irradiation.

Dans le cadre de l'irradiation de combustibles MOX de différentes microstructures avancées (expérience MORGANE), les taux de combustion visés sont de 60 à 70 GWJ/tU, soit environ 20 cycles d'irradiation.

Dans le paragraphe suivant nous verrons la modélisation neutronique actuellement réa-

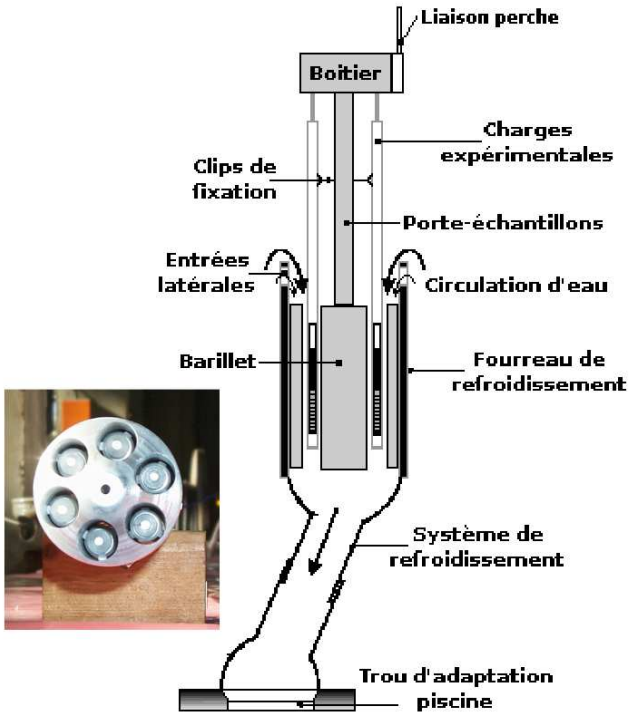


FIG. 8.1 – Présentation du dispositif TANOXOS

lisée par l'exploitant du réacteur OSIRIS.

8.2 Modélisation actuelle

A la fin d'une expérience d'irradiation, il est important de bien maîtriser le taux de combustion atteint par chaque crayon combustible. Ce dernier est obtenu en intégrant la puissance de fission dégagée par chaque crayon sur la durée de l'expérience.

Les calculs neutroniques ont pour but de déterminer la puissance absolue (en W) sur chaque crayon en fonction des courants collectrons rhodium instrumentant le dispositif. La puissance linéique est un résultat direct des calculs neutroniques et dépend de la puissance de fonctionnement du réacteur durant l'irradiation ainsi que de la distance du dispositif par rapport au cœur.

Avec la mesure du courant collectron, il est possible d'obtenir la puissance linéique mesurée de manière indirecte et donc le taux de combustion atteint par chaque crayon connaissant la masse linéique m_ρ d'uranium métal du crayon.

Afin de garantir que l'expérience se déroule dans les conditions REP, le pilotage est assuré par des mesures de température à l'aide de thermocouples positionnés dans une partie évidée de crayon combustible.

Les calculs neutroniques actuels sont réalisés avec la méthode des probabilités de première collision P_{ij} ([Reu03]), à 99 groupes, en géométrie générale. Cette méthode permet de définir la géométrie la plus réelle possible pour l'expérience et son environnement. Cependant, cette méthode est coûteuse en temps de calcul pour un nombre de régions

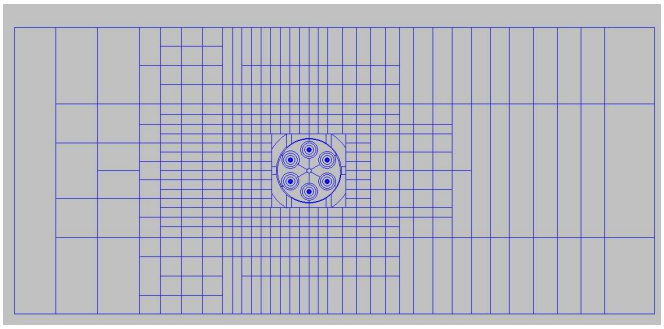


FIG. 8.2 – Maillage en régions de flux plat du dispositif TANOXOS

de flux plat élevé. C'est pourquoi seule une partie de la géométrie est représentée par le maillage en régions de flux plat de la figure 8.2 et le cœur du réacteur est représenté de manière homogène. De plus, la méthode des P_{ij} résout l'équation intégrale du transport des neutrons, et ne permet pas de prendre en compte la dépendance angulaire du flux de neutrons. L'anisotropie de la loi de choc des neutrons sur l'hydrogène est partiellement prise en compte par l'intermédiaire d'une correction sur les sections efficaces concernées.

Les calculs thermohydrauliques ont été réalisés à la conception du dispositif et ont permis de qualifier le comportement du dispositif. Ils ne sont pas à nouveau réalisés à chaque calcul neutronique.

Les calculs traitant du comportement du combustible sont réalisés avec le code METEOR, en introduisant l'historique d'irradiation en entrée du calcul.

8.3 Présentation du couplage envisagé

Le développement de la plate-forme de simulation des dispositifs expérimentaux est également l'occasion de mettre en œuvre et de coupler les outils de simulation les plus modernes et les plus performants pour améliorer la précision des calculs numériques. Le dispositif TANOXOS se prête bien à cet exercice, notamment pour la neutronique où la modélisation précise des flux neutroniques en réflecteur est parfois délicate, d'autant plus si le dispositif est relativement éloigné du cœur.

Les développements récents de la méthode des caractéristiques (MOC) dans APOLLO2 permettent d'envisager des calculs 2D cœur complet, ce qui peut à terme favoriser l'unification des schémas de calcul pour les dispositifs expérimentaux, qui sont relativement dépendants de la position du dispositif dans le réacteur.

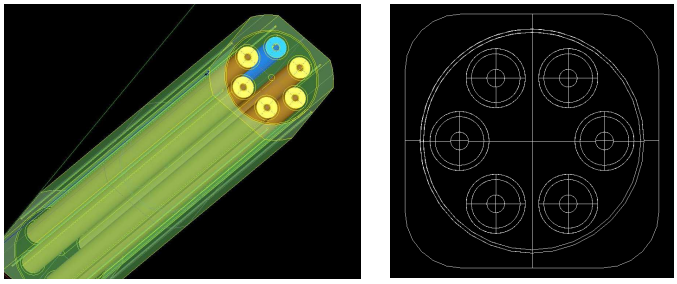
L'application ALCYONE du nouveau code de calcul PLEIADES ([HVP05]), mise en œuvre au CEA, réalise des simulations du comportement d'un crayon combustible REP sous irradiation. Le dispositif TANOXOS étant dédié à l'expérimentation sous irradiation de ces mêmes crayons, c'est cette application ALCYONE qui a été choisie pour réaliser les calculs de comportement du combustible. Les versions d'ALCYONE, mises à disposition pour la thèse, permettent de réaliser des calculs 1D renseignant sur la

température du crayon en différents points (nœuds) placés sur un rayon et leurs déplacements (translations) induits par des contraintes mécaniques.

Le couplage envisagé consiste à réaliser un calcul neutronique 2D du cœur complet du réacteur OSIRIS avec le solveur MOC du code APOLLO2, et le formulaire de calcul ANUBIS [SBd+07] couplé à six calculs comportement combustible des six crayons REP de TANOXOS avec l'application ALCYONE. Le paragraphe suivant présente la modélisation par Entités Technologiques du dispositif TANOXOS. Des précisions sur la simulation neutronique proposée et la simulation du comportement combustible seront ensuite apportées. Le paragraphe 8.3.4 présente le couplage envisagé.

8.3.1 Modèle d'Entités Technologiques

Le modèle d'Entités Technologiques qui décrit le dispositif TANOXOS a été rédigé dans un fichier PYTHON, disponible en annexe K. La représentation 3D et une coupe en 2D, réalisées à partir de ce modèle, sont présentées sur les figures 8.3(a) et 8.3(b).



(a) Vue en 3D (b) Vue d'une coupe en 2D

FIG. 8.3 – Présentation du dispositif TANOXOS modélisé par l'arbre d'Entités Technologiques

La racine de l'arbre d'Entités Technologiques doit être un objet qui englobe la totalité des objets du dispositif. Ainsi, l'Entité technologique à la racine de l'arbre est constituée par le Fourreau. Ce Fourreau contient un cylindre d'eau (voir figure 8.4), et il est donc possible de modifier l'épaisseur du Fourreau en faisant varier le rayon de ce cylindre.

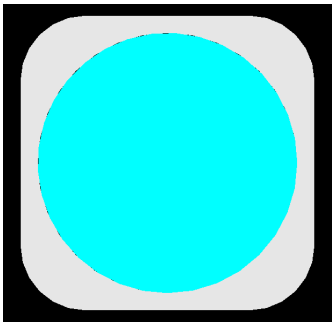
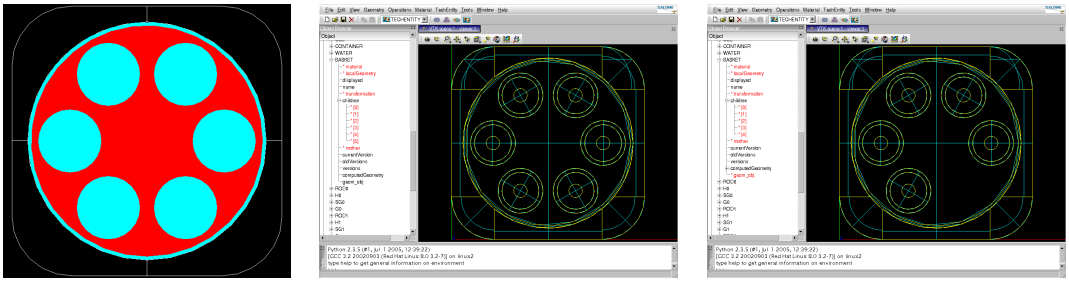


FIG. 8.4 – Le Foureau contient un cylindre d'eau.

Le barillet qui contiendra les six crayons combustibles, est plongé dans l'eau du fourreau (figure 8.5(a)). Ce barillet est modélisé par un cylindre dont le rayon peut être modifié.

Les six trous sont construits par les six cylindres d'eau destinés à accueillir les crayons. Ainsi, les tailles et le nombre de trous peuvent aussi être paramétrés à travers la liste de `children` de l'Entité Technologique correspondant au barillet. Par exemple, sur le modèle d'Entités Technologiques réalisé en annexe K, le simple retrait d'un sixième trou d'eau dans la liste `children` du barillet, change la représentation de TANOXOS vue en figure 8.5(b) en la représentation vue en figure 8.5(c).



(a) Barillet contenant les six crayons (b) Barillet à six crayons (c) Barillet à cinq crayons

FIG. 8.5 – Barillet du dispositif TANOXOS et démonstration du paramétrage du nombre de crayons.

Les crayons entourés de leurs gaines, plongés dans les trous d'eau du barillet se trouvent à l'intérieur d'une surgaine.

8.3.2 Modélisation Neutronique du dispositif

Alors que dans la modélisation neutronique présentée au paragraphe 8.2, seul le dispositif est modélisé et le cœur est remplacé par un milieu simulant le comportement moyen du cœur, la modélisation neutronique proposée pendant la thèse consiste à modéliser l'ensemble du cœur OSIRIS ainsi que le dispositif. Cette modélisation est issue d'une adaptation du formulaire de calcul neutronique ANUBIS.

Le formulaire ANUBIS permet de simuler le comportement neutronique du réacteur OSIRIS en évolution, il est composé de deux schémas de calcul : un schéma dit d'exploitation qui réalise des calculs 3D homogènes en des temps de calcul courts et un schéma dit de référence qui simule les hétérogénéités du cœur à 2D, avec une contrainte moindre sur le temps de calcul. L'adaptation réalisée pendant la thèse est techniquement détaillée au chapitre 9. Elle a consisté à permettre d'insérer le dispositif TANOXOS dans le réflecteur du cœur d'OSIRIS pour le schéma de référence. Les différentes étapes sont les suivantes :

- L'exportation d'une coupe du dispositif en 2 dimensions, à partir de l'arbre d'Entités Technologiques ;
- L'importation de la coupe du dispositif et des milieux associés dans l'outil de maillage neutronique SILENE ([Sta97]) et l'insertion dans le maillage de calcul du cœur OSIRIS ;
- La prise en compte dans le formulaire ANUBIS, des différents milieux du dispositif TANOXOS ;

– L'exportation des résultats et leurs post-traitements pour accéder aux grandeurs d'intérêt nécessaires à l'étude de l'expérience d'irradiation.

Le calcul neutronique, résout l'équation de Boltzmann pour les neutrons (voir paragraphe 2.2.1) en 2D. Les résultats[1] extraits de ce calcul sont en particulier les flux neutroniques par groupes énergétiques et les puissances neutroniques.

L'ensemble des données d'entrées de la neutronique comporte donc la géométrie du domaine d'étude et le maillage spatial et énergétique (découpage en groupes), la composition isotopique et la température de chacun des milieux.

Le calcul d'évolution permet de calculer des nouvelles compositions isotopiques des milieux combustibles selon un pas de Burnup[2] (combustion) donné.

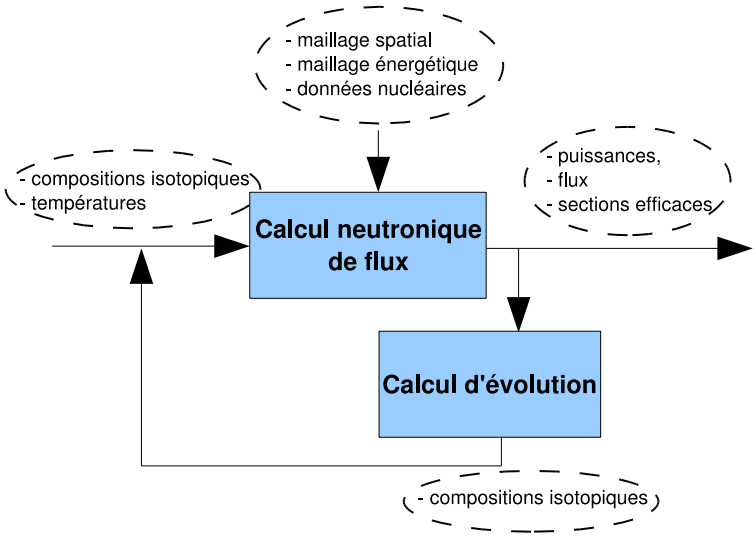


FIG. 8.6 – Schéma simplifié du calcul neutronique (calcul d'évolution quasi sationnaire)

La figure 8.6 présente un schéma simplifié du calcul neutronique et laisse apparaître les différentes données échangées.

8.3.3 Modélisation du comportement des crayons du dispositif

Le projet de co-développement PLEIADES a pour objectif le développement d'une plate-forme de simulation du comportement des combustibles de toutes filières nucléaires. Un des axes de développement de cette plate-forme est la mise en œuvre d'applications permettant de capitaliser et de mutualiser l'ensemble des connaissances physiques et techniques du comportement des combustibles de toutes les fillières nucléaires. ALCYONE est une des premières applications de ce projet, elle est dédiée à l'étude du comportement des combustibles REP.

[1] Le schéma de calcul permet par ailleurs de réaliser des traitements particuliers sur les données d'entrées du schéma de calcul telles que les données nucléaires de base.

[2] Quantité ayant une dimension temporelle mesurée en $\frac{MWjours}{tonne}$.

La version de l'application ALCYONE utilisée pour la thèse permet de simuler le comportement d'une pastille d'un crayon REP sous irradiation, de sa gaine et du gap pastille gaine en dimension 1 (un rayon du crayon). Ces calculs sont effectués en considérant différents modèles physiques de plus bas niveau, dont un modèle simplifié de neutronique et un modèle de thermohydraulique.

Le modèle de neutronique (appelé modèle *RADAR*) fournit à l'application ALCYONE des résultats relatifs à la distribution de puissance neutronique dans le crayon, des flux neutroniques et des sections efficaces.

Le modèle de thermohydraulique fournit quant à lui une température de paroi et un coefficient d'échange entre le fluide caloporteur et la gaine du crayon.

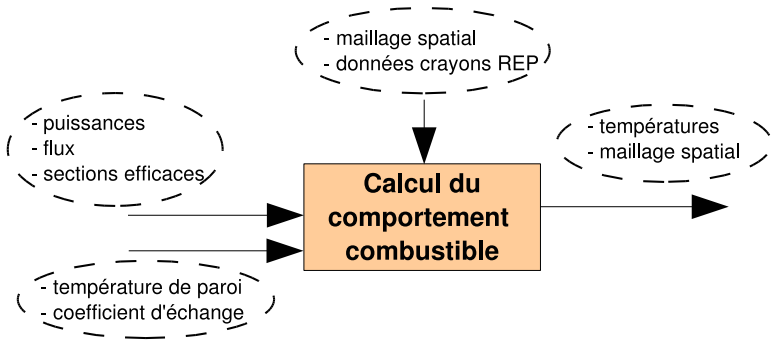


FIG. 8.7 – Schéma simplifié du calcul du comportement d'un crayon REP

Les grandeurs d'intérêt issues de la simulation du comportement combustible (dont un schéma simplifié est présenté sur la figure 8.7) sont nombreuses. Nous nous intéressons dans cette étude à la distribution de température et aux déformations de la pastille et de la gaine.

8.3.4 Couplage envisagé

En neutronique, la simulation proposée permet d'avoir une grande précision sur les flux, les sections efficaces, la puissance neutronique et l'évolution des compositions isotopiques. L'évaluation de la température des milieux des six crayons combustibles de TANOXOS en entrée de la neutronique est calculée par l'application ALCYONE. Le modèle RADAR est remplacé par les résultats issus de la neutronique.

Que ce soient la simulation neutronique du cœur, ou bien les simulations des comportements de crayons REP, toutes sont dépendantes du temps, selon un schéma en temps qui leur est propre. Il a donc été choisi d'encapsuler ces simulations dans des interfaces `UnSteadyProblem` (voir paragraphe 7.1.5).

La figure 8.8, présente le schéma du couplage réalisé. Le couplage complet consisterait à introduire dans ce schéma un calcul de thermohydraulique pour déduire la température de l'eau qui impacte les températures des parois des gaines des crayons. Techniquement, cela consiste à remplacer le modèle de thermohydraulique simplifié proposé par

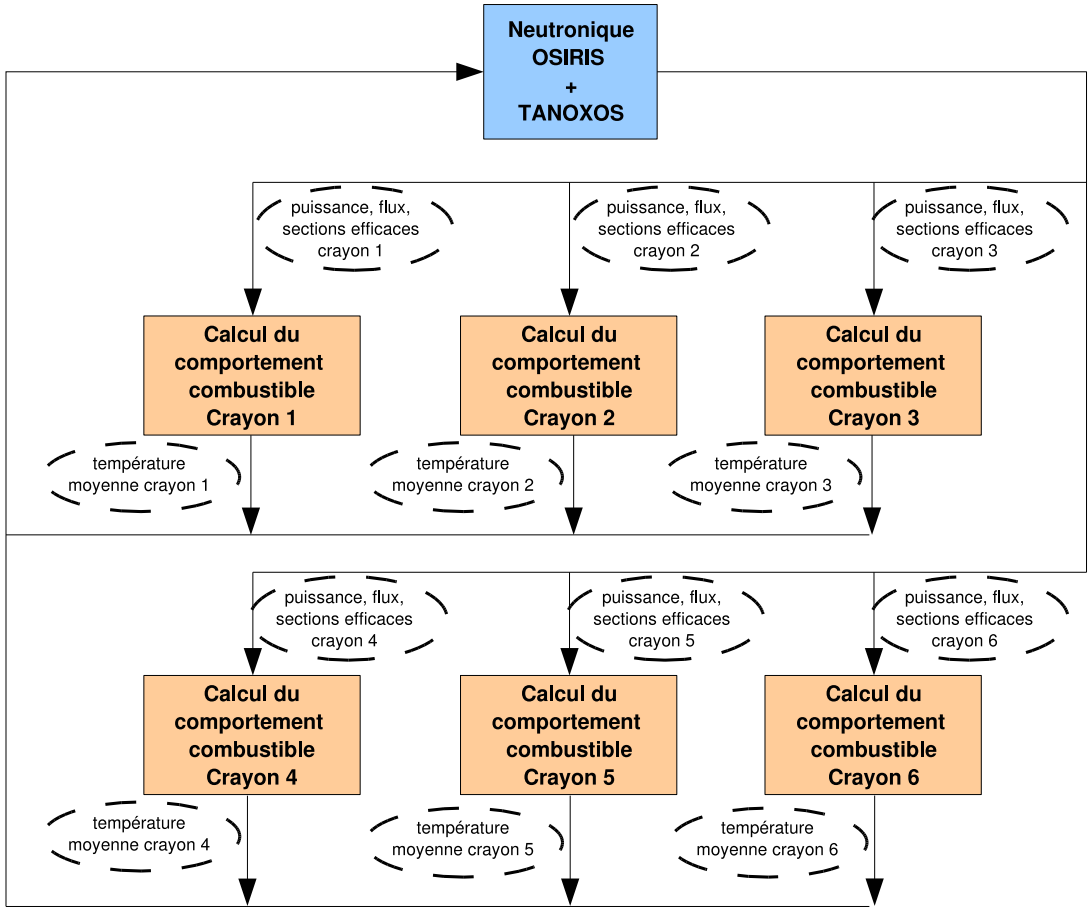


FIG. 8.8 – Schéma du couplage neutronique / comportement combustible

ALCYONE, par un modèle plus précis.

Les solutions aux prinicipales difficultés techniques révélées dans la mise en œuvre d'un modèle de thermohydraulique, réalisé par le code TRIO_U, ont été développées dans ces travaux de thèse. Le problème considéré est un problème couplé de conduction-diffusion dans le solide (gaine et combustible) et d'écoulement monophasique dans le liquide (eau). L'objectif est alors de simuler l'écoulement du fluide caloporteur proche des crayons du dispositif TANOXOS, afin d'en déduire une température de paroi (gaine) plus précise que le modèle de thermohydraulique simplifié fourni par l'application ALCYONE. Pour résoudre ce problème, une des complexités provient de la nécessité d'un maillage conforme sur la paroi d'échange des domaines liquide et solide. Ce maillage doit être réalisé automatiquement à partir du modèle d'Entités Technologiques, qui doit donc repérer cette surface d'échange dans la géométrie. Un des grand intérêts de l'arbre de construction du modèle d'Entités Technologiques est qu'il permet de repérer aisément cette surface d'échange. Le calcul de thermohydraulique permet ainsi d'obtenir une température de paroi introduite dans le calcul de comportement combustible (ainsi qu'une température moyenne fluide qui pourra être réintroduite dans le calcul de neutronique à l'itération suivante). Toutefois, les contraintes de temps liées à la thèse n'ont pas permis d'aboutir ce modèles. Les variations de température combustible dû à cette précision de calcul n'ont cependant qu'une influence limitée sur les résultats neutroniques.

L'intérêt du couplage avec ALCYONE est double. D'une part il ne nécessitait pas de temps de mise en œuvre trop important, les schémas de calcul ayant déjà été réalisés au DEC[3]. D'autre part, il pourra permettre, à terme, de vérifier rapidement la capacité des Entités Technologiques à gérer les changements de géométrie au cours de l'évolution du combustible. Ce changement de géométrie influe directement sur le rapport de modération qui a un impact important en neutronique. Dans le cadre de cette thèse, l'objectif est plutôt de mettre en pratique les concepts développés dans la seconde partie. Le temps nécessaire au couplage complet n'étant pas disponible.

Le chapitre suivant présente en détail le calcul de neutronique et les développements qui ont permis d'encapsuler le formulaire ANUBIS, traitant le dispositif TANOXOS, dans la plate-forme de modélisation des dispositifs expérimentaux.

[3]DEC : Département d'Etudes de Combustible du CEA de Cadarache

Chapitre 9

Etude Neutronique

9.1 Modèle physique

Le calcul neutronique est réalisé à 2D avec la méthode des caractéristiques du code APOLLO2 (version 2.8). Pour valider le schéma de calcul déterministe de TANOXOS, on réalise un calcul Monte-Carlo cœur neuf avec TRIPOLI4 (voir chapitre 2). Les résultats de la validation sont présentés en annexe L.

Le maillage utilisé pour le calcul de cœur est représenté sur la figure 9.1. Ce maillage est construit par étapes, le chargement final du cœur pouvant varier d'un calcul à l'autre. Dans cette étude, on représente un cœur disposant de 5 boîtes à eau 4 trous, et 3 barres de commande (numéro 3, 4 et 6) insérées dans le cœur. Ce maillage dispose d'un espace libre pour insérer le dispositif créé par les entités technologiques. Les assemblages combustibles situés en première et dernière colonne du cœur sont maillés plus finement pour bien reproduire la remontée du flux thermique au voisinage des réflecteurs. Pour conserver un nombre de mailles de l'ordre de 30000, il est nécessaire d'homogénéiser l'âme combustible dans sa gaine. Cette étape est réalisée au moment de la pondération flux/volume des sections efficaces pour passer de 172 groupes à 20 macro-groupes par l'intermédiaire d'un calcul de flux à 1D sur une traverse de l'assemblage combustible.

Les paramètres d'intégration pour le tracé des caractéristiques sont donnés par $\Delta R = 0.05$ et $N\Phi = 12$, ce qui représente un pas de 0.5 mm et 12 directions (d'angle de rotation entre 0 et π radians) imposées à l'ensemble des trajectoires. Pour les calculs en évolution, on dénombre 88 milieux combustibles évoluants (4 par plaques), et 8 milieux composés de poisons consommables en aluminium boré évoluant par assemblage standard. Les milieux panier de contrôle absorbant en hafnium sont séparés en 3 couches d'épaisseurs respectives 0.3 mm, 2.4 mm et 0.3 mm pour bien distinguer les effets d'autoprotection (effet de peau) sur l'épaisseur de l'absorbant.

La répartition de flux thermique sur le cœur est présentée sur la figure 9.2. Dans ce calcul, le cœur est entièrement chargé de combustible neuf. Les crayons du dispositif TANOXOS sont en UO_2 d'enrichissement massique $e = 7\%$. Le dispositif est placé à 90 mm du caisson nord d'OSIRIS.

Le schéma de calcul prend en compte l'évolution des matériaux combustibles et permet

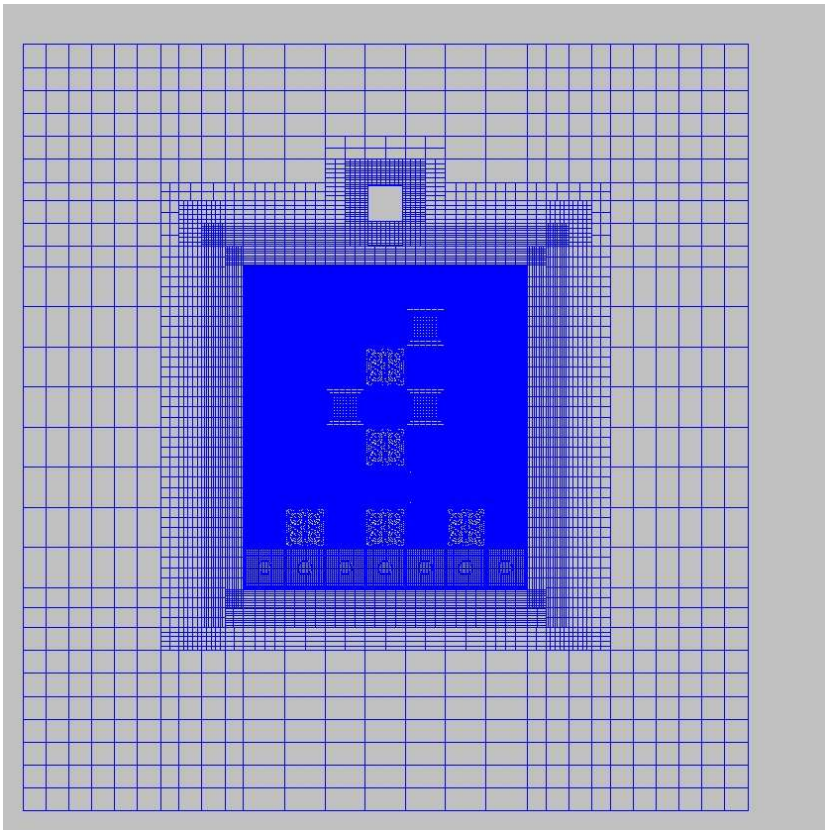


FIG. 9.1 – Maillage en régions de flux plat du cœur d'OSIRIS

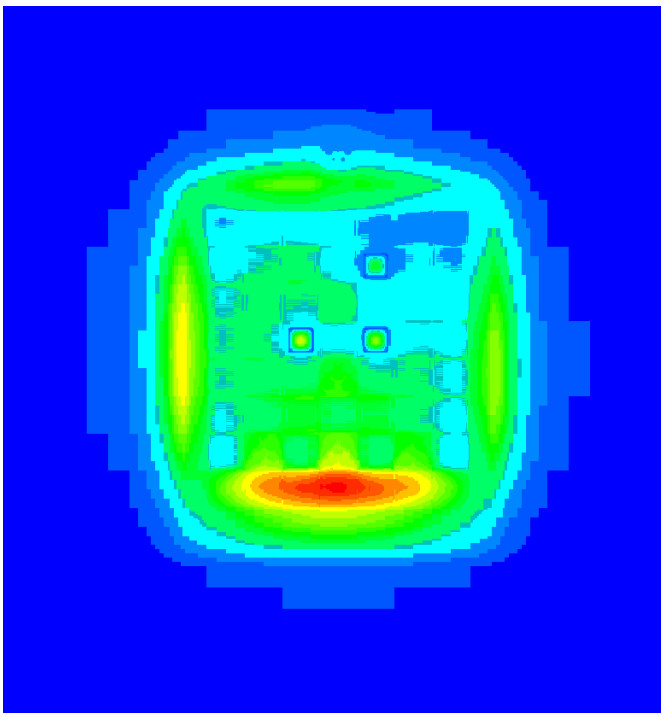


FIG. 9.2 – Représentation par SILENE du flux en neutrons thermiques calculé sur le cœur d'OSIRIS neuf avec le dispositif TANOXOS (unité arbitraire).

de charger aussi bien des cœurs neufs que des cœurs rechargés, tels que celui du cycle 192[1] présenté par la figure 9.3.

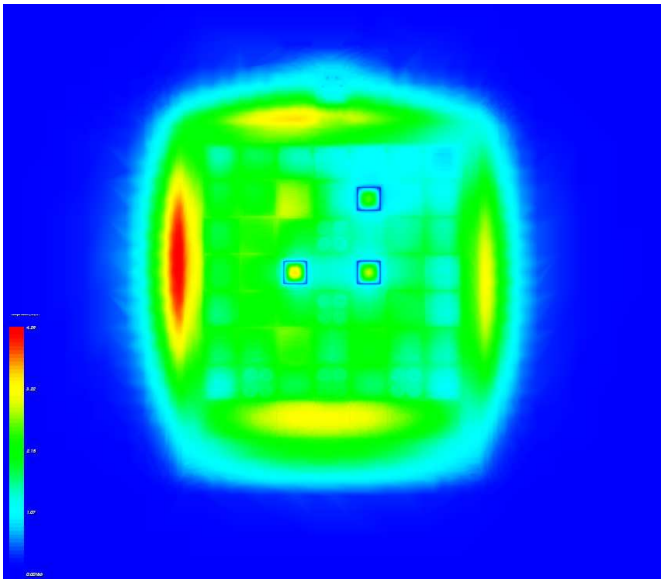


FIG. 9.3 – Représentation par SALOME (développé au paragraphe 9.2.2) du Flux en neutrons thermiques calculé sur le cœur d'OSIRIS au cycle 192 avec le dispositif TANOXOS (unité arbitraire).

Les deux représentations du flux thermique dans OSIRIS avec des chargements différents (figures 9.2 et 9.3) permettent de constater l'impact du chargement sur l'évolution des paramètres neutroniques du cœur mais aussi du dispositif TANOXOS à proximité de celui-ci (variations importantes du niveau de flux en réflecteur).

9.2 Intégration

L'intégration dans SALOME consiste à développer une classe spécifique, qui implémente l'interface `UnSteadyProblem`. Outre le développement des méthodes de cette interface, deux avancées techniques ont été nécessaires pour permettre les échanges de données entre les Entités Technologiques et ANUBIS et ceux entre ANUBIS et TECHENTITY (figure 9.4).

Les échanges de données entre le modèle d'Entités Technologiques et ANUBIS consistent en un pré-traitement des données, présenté au paragraphe suivant. La principale difficulté de cette étape a été d'échanger les géométries entre le modèle de TANOXOS et le mailleur SILENE. Le paragraphe 9.2.2 présente le post-traitement des données nécessaire à l'échange d'ANUBIS et TECHENTITY.

9.2.1 Pré-traitement

Dans ce paragraphe, il sera développé les solutions techniques qui nous ont permis de convertir le modèle STEP en modèle SILENE.

[1] Le cycle 192 est pris arbitrairement pour valider et qualifier les résultats de neutronique

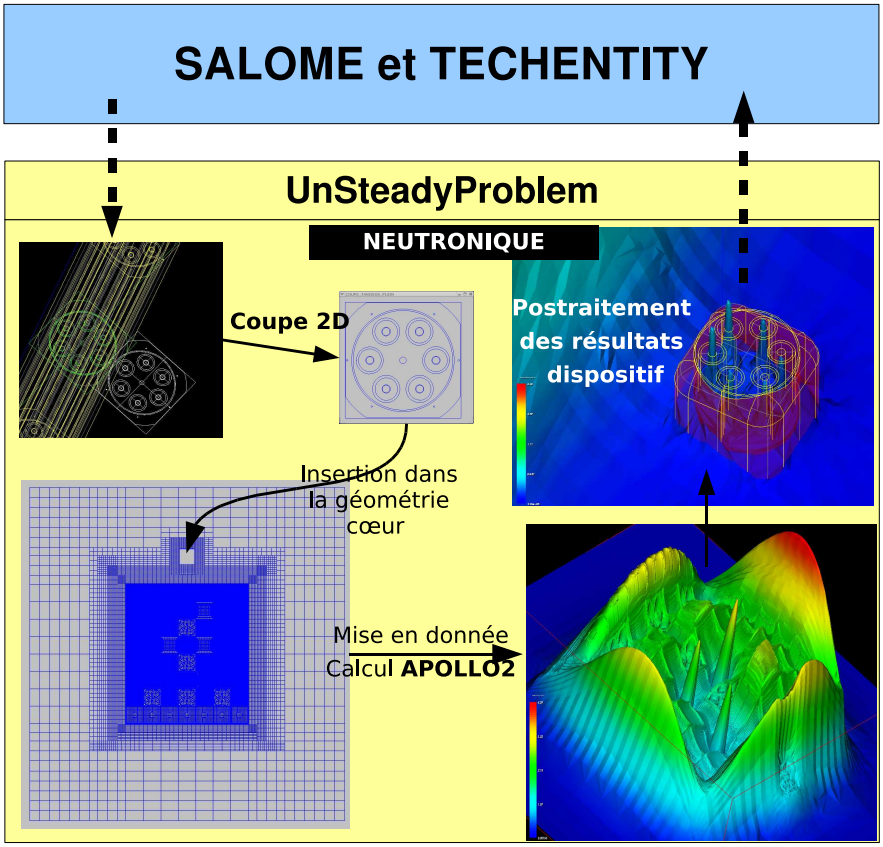


FIG. 9.4 – Schéma d'intégration du calcul neutronique

Le modèle géométrique de SILENE

Une terminologie, détaillée ci-dessous, est associée au modèle de donnée interne de SILENE :

- *Motif* : ensemble des données décrivant une géométrie. Le motif est décrit par des ensembles d'objets représentant les nœuds (points), équations (segments, cercles, arcs et développantes de cercles), mailles, références à d'autres motifs, niveaux en z (3D), régions (pour le calcul de flux), macro-régions (pour le calcul des courants d'interface), milieux, zones de sortie et conditions aux limites ;
- *Nœuds* : point dans l'espace (X, Y) ;
- *Equation* : définie par 4 points au plus. Il existe 4 spécialisations des équations : segment, arc de cercle, cercle, développante ;
- *Maille* : forme géométrique définie par un ensemble d'équations fermées ;
- *Périmètre* : ensemble d'équations définissant un périmètre fermé (maille et motif) ;
- *Région* : ensemble de mailles sur plusieurs niveaux (z) ;
- *Milieu* : ensemble de régions sur plusieurs niveaux ;
- *Macro* : ensemble de régions contiguës, spécifique à Apollo2 ;
- *Sortie* : ensembles quelconques de régions.

La représentation d'une géométrie SILENE est de type B-Rep (voir paragraphe 3.1). La figure 9.5 permet de comprendre comment la géométrie est construite dans SILENE.

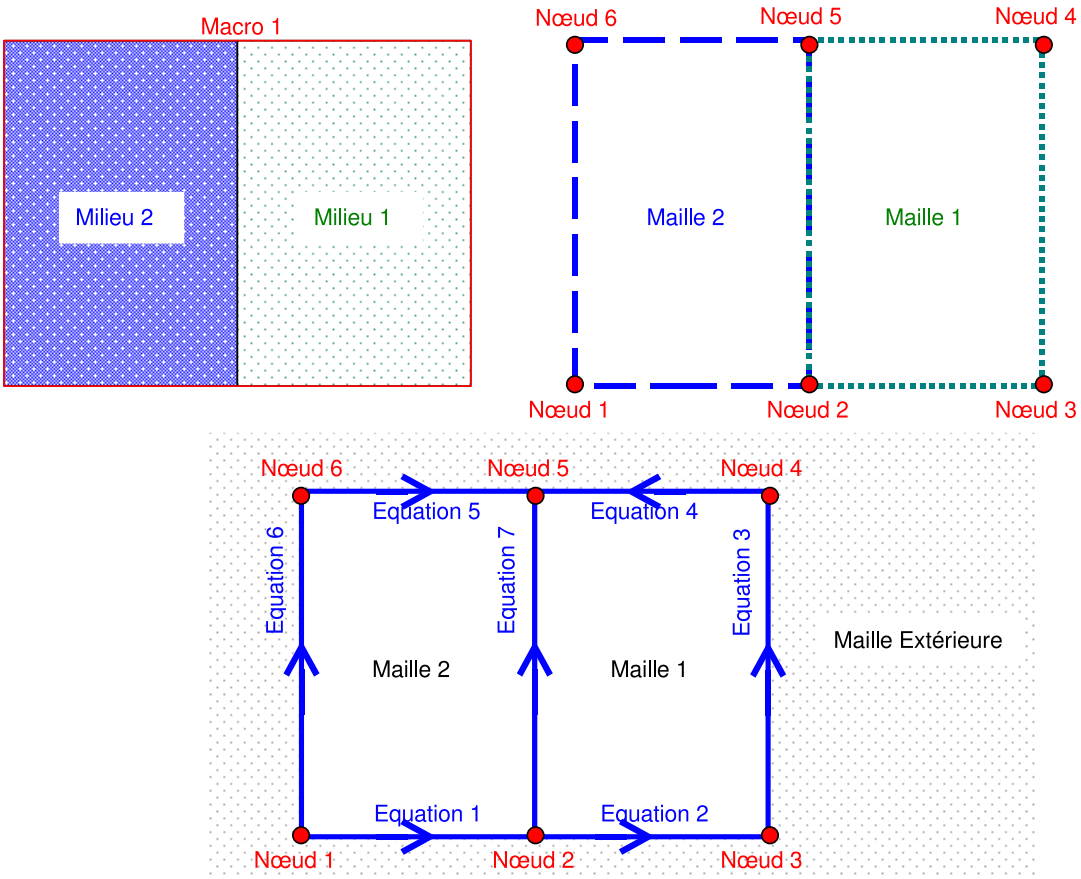


FIG. 9.5 – Exemple de modélisation d'objets géométriques SILENE

En considérant que la géométrie est définie sur un seul niveau (géométrie 2D), une maille est associée à un milieu qui représente un matériau pour le code de calcul. Une maille possède un périmètre qui est une agrégation d'équations (de bords). Dans l'exemple de la figure 9.5, les mailles 1 et 2 ont, dans leur périmètre, l'équation 7 en commun. Cette équation n'est pas dupliquée. La maille extérieure est une maille particulière, en ce sens qu'il n'y a pas de milieu qui lui est associé.

Chaque équation a un sens de parcours. Par exemple l'équation 1 va du nœud 1 vers le nœud 2. Ce sens est nécessaire pour conserver l'information topologique de la géométrie. A partir de ce sens il est possible de déduire un côté droit et un côté gauche d'une équation. Chaque équation fait référence à deux mailles dans deux attributs distincts : la maille de gauche et la maille de droite. Pour l'équation 7, la maille de gauche est la maille 2, la maille de droite est la maille 1 ; tandis que pour l'équation 3, la maille de gauche est la maille 1 et la maille de droite est la maille extérieure. Ces informations permettent de faire la différence entre l'intérieur et l'extérieur d'une maille.

Pour convertir la géométrie du modèle STEP (de plus amples détails sur la norme STEP et la protocole d'application 214 sont donnés en annexe C) vers le modèle SILENE nous avons considéré les équivalences suivantes :

– Une instance de la classe `Advanced_face` équivaut à une instance de Maille ;

– Une instance de la classe `Edge_curve` équivaut à une instance de Equation ;
– Une instance de la classe `Cartesian_point` équivaut à une instance de Nœud.

Elimination des redondances

Comme le montre la figure 9.5 dans la représentation SILENE, à la jointure de deux mailles il n'y a qu'une seule instance de la classe Equation qui persiste (par exemple pour la maille 1 et 2 il s'agit de l'Equation 7). Dans le cas de la représentation STEP de la géométrie (figure 9.6), deux instances Edge_curve existent. De plus ces deux instances de Edge_curve, s'appuient sur des instances de points (ou nœuds) de départ et d'arrivée différentes. Ces points possèdent des coordonnées identiques.

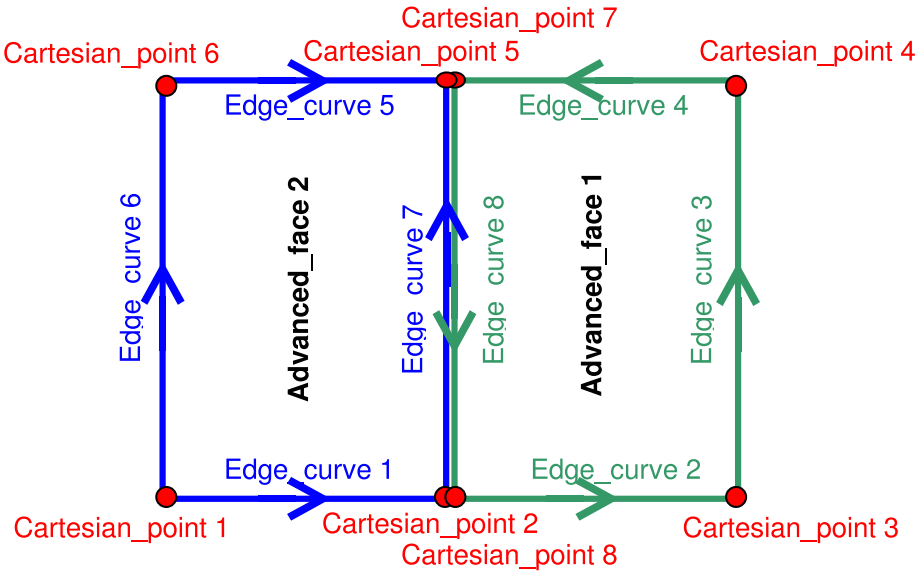


FIG. 9.6 – Représentation STEP équivalente à la figure 9.5

Pour passer d'une représentation STEP à une représentation SILENE, il est nécessaire de repérer les instances de Edge_curve (Equation) et Cartesian_point (Nœud) identiques afin d'éliminer les redondances, et ensuite de créer les instances Equation et Nœud correspondantes dans la représentation SILENE.

Reconstruction d'une maille

Comme il l'a été présenté plus haut, une maille est principalement définie par son périmètre et par son sens de parcours qui permet d'indiquer l'espace intérieur et l'espace extérieur de la maille.

En considérant le principe selon lequel le périmètre extérieur d'une maille est parcouru dans le sens direct, alors l'intérieur de la maille se trouve à la gauche du périmètre. Pour les périmètres intérieurs parcourus dans le sens direct, l'intérieur de la maille se trouve à leur droite[2] (figure 9.7).

[2] La convention choisie n'est pas classique (en général les périmètres intérieurs sont parcourus dans

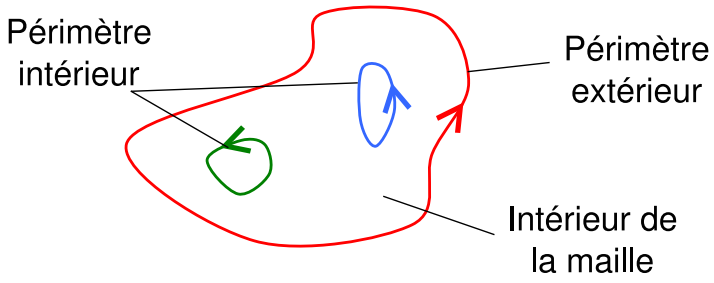


FIG. 9.7 – Sens de parcours et périmètre d'une maille

D'après le théorème de Green, on peut déduire que si l'on parcourt le périmètre Γ d'une maille dans le sens direct alors la relation 9.1 (basée sur le théorème de Stokes) est vraie et permet de calculer l'aire à l'intérieur de la maille.

$$\oint_{\Gamma} -y\,dx = Aire\ de\ la\ maille \tag{9.1}$$

En parcourant le périmètre dans le sens indirect, alors la relation vue au-dessus (de sens de parcours opposé) nous donne l'aire à l'intérieur de la maille mais de signe opposé.

La représentation STEP de la géométrie ne permet pas d'obtenir directement le sens de parcours d'une maille, et donc ne permet pas de différencier la maille gauche de la maille droite pour les instances de la classe Equation.

En effet, le format STEP permet de disposer aisément de l'ensemble des bords qui constituent le périmètre d'une maille. Cependant, ceux-ci sont désordonnés. Dans un premier temps ces bords sont ordonnés suivant un sens arbitraire. Ensuite un calcul d'aire avec la relation 9.1 permet de changer le sens de parcours, si le signe de l'aire est négatif (voir figure 9.8).

Connaissant l'ordre des bords du périmètre pour que celui-ci soit parcouru dans le sens direct, il est maintenant possible de distinguer la maille droite et la maille gauche de chaque instance de la classe Equation.

Les instances des classes Nœud, d'Equation et Maille peuvent être construites et ajoutées au motif. L'utilisation des fonctionnalités implémentées dans SILENE pendant la thèse est détaillée en annexe M.

9.2.2 Post-traitement

Afin de rendre compatible les résultats de neutronique avec SALOME, il est nécessaire de mettre en œuvre les moyens techniques permettant l'exportation des résultats de calcul au format MED de SALOME.

---

le sens indirect), elle permet cependant d'utiliser les mêmes algorithmes pour reconstruire et orienter les mailles.

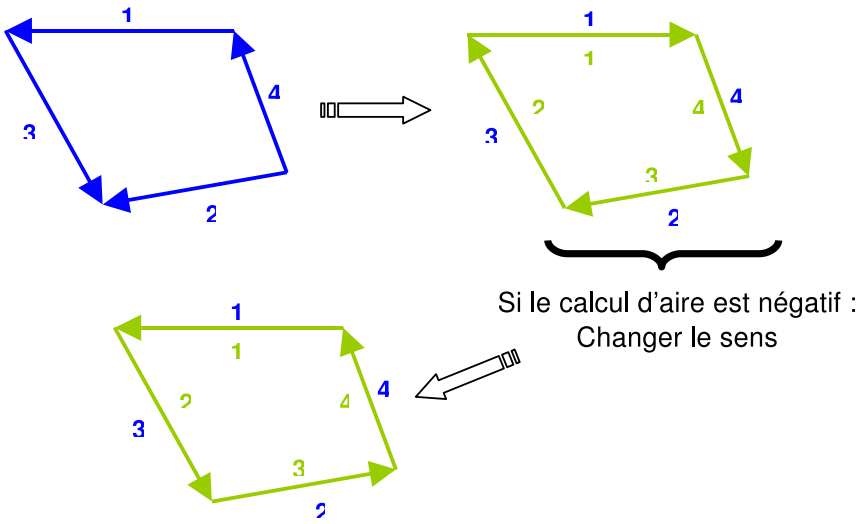


FIG. 9.8 – Orientation d'une maille

Lors d'un calcul APOLLO2 MOC, l'outil SILENE de découpe de motif géométrique en région de flux plat est utilisé. Cet outil enregistre ses modèles dans un format de fichier spécifique en extension *.dat*. Ce format de fichier décrit des nœuds (points dans l'espace en 2D), des équations[3] de type segments, arcs de cercles, développantes de cercles et cercles, des régions associées à des équations (périmètres), des milieux associés à des régions et des macros.

Le code de calcul APOLLO2 utilise le format de fichier *TDT* (format spécifique au code APOLLO2) pour réaliser des calculs de flux avec le solveur MOC. Les fichiers au format *TDT* sont générés par SILENE à partir d'un motif décrit dans un fichier *.dat*.

Le module GIBIANE *POSTSIL :* du code de calcul APOLLO2, génère un fichier en extension *.res* contenant des résultats de calcul du solveur MOC. Ce fichier contient les noms de milieux, les sections efficaces par milieu, les flux par région de flux plat et par groupe, les volumes des régions (sur une hauteur unité).

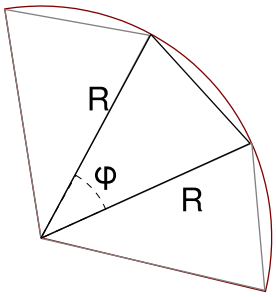
Le format MED est un modèle de données représentant des maillages de types différences finies, éléments finis, volumes finis. Des interfaces en C et en Fortran sont mises à disposition pour lire et écrire des données dans ce format. Nous avons utilisé l'interface PYTHON de SALOME (Version 2.2.8 à 3.2.6) en cours de développement. Cette interface se présente sous la forme d'un module PYTHON permettant de construire en mémoire et d'écrire dans un fichier des maillages et des champs de valeurs au format MED. Les versions de SALOME utilisées durant la thèse et du composant de visualisation scientifique VISU, permettent de traiter des fichiers MED contenant des maillages non-structurés composés d'entités géométriques de type quadrangle et triangle pour la 2D. Les entités de bords courbes et les polygones quelconques, rencontrés en neutroniques, ne sont pas supportés.

[3] connectivités entre nœuds, elles représentent des bords de régions

En fonction de cela et parce que les problèmes d'interpolations de grandeurs sur des triangles sont largement traités dans la littérature adaptée ([FG99]), il a été choisi d'exporter le modèle de SILENE vers le format MED en le discrétisant en triangles. Un modèle de données intermédiaire a donc été développé.

Discrétisation

Les arcs de cercles sont discrétisés en segments. Le critère de discrétisation est l'écart relatif entre l'aire de la portion de disque S_{arc} et l'aire du polygone correspondant à la discrétisation de cette portion de disque. Cela conduit à quantifier l'écart relatif entre un arc de cercle et le triangle formé avec la corde de l'arc.



On a $S_{arc} = \frac{\varphi \cdot R^2}{2}$ et l'aire du triangle isocèle $S_{tri} = \frac{R^2 \sin\varphi}{2}$;

d'où

$$\frac{S_{arc} - S_{tri}}{S_{arc}} = 1 - \frac{\sin\varphi}{\varphi}$$

On peut alors constater que l'écart relatif ne dépend que de l'angle de discrétisation. Cet angle φ est calculé une fois pour toutes en début d'exportation d'un motif en fonction du pourcentage d'écart d'aires choisi par l'utilisateur.

Les arcs de cercles sont discrétisés, les développantes de cercles ne sont pas prises en compte. La méthode utilisée ne permet pas de considérer les cercles lors de la triangulation. Les motifs possédants des cercles sont donc découpés en arcs de cercles avant le calcul MOC pour pouvoir être exportés au format MED.

L'objectif de ce maillage est de fournir une simple représentation en triangles de la géométrie de calcul neutronique, il n'y a pas de contrainte quant à la qualité du maillage. Ainsi, il a été privilégié d'implémenter une méthode rapide à mettre en œuvre et dont les performances en termes de temps de calcul sont convenables au vu du temps de calcul d'un pas de temps en neutronique (de l'ordre d'une demi-heure par pas de temps sur un processeur équivalent Xéon 3,6GHz).

Application

Le développement de cet outil de post-traitement est suffisamment général pour être appliqué à n'importe quels résultats de calculs réalisés avec le solveur MOC. Ainsi, cet outil a pu être utilisé avec le formulaire HORUS3D traitant de la neutronique du RJH et le formulaire ANUBIS (voir figure 9.9).

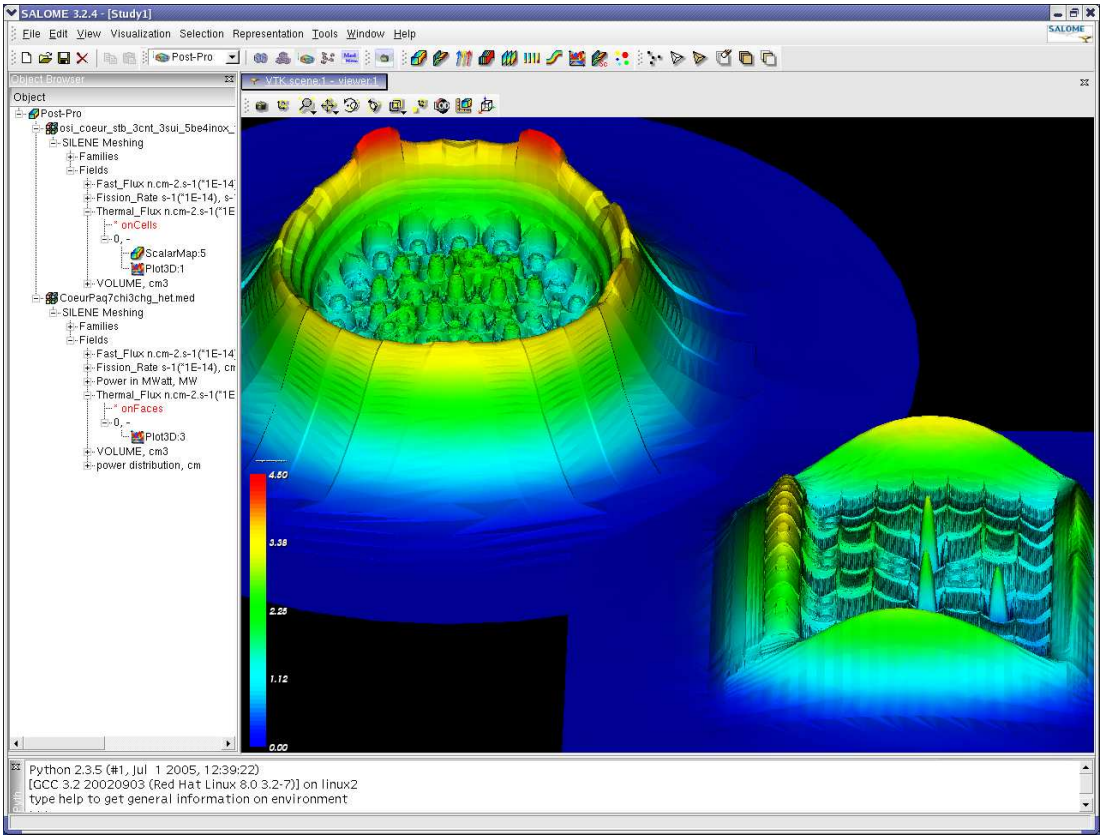


FIG. 9.9 – Représentation des flux en neutrons thermiques dans SALOME du RJH et du réacteur OSIRIS

Chapitre 10

Séquencement des disciplines

10.1 Encapsulation de la neutronique

La simulation de neutronique présentée au paragraphe 9 permet de réaliser des calculs d'évolution[1]. De ce fait, il ne s'agit pas d'un problème stationnaire et il a donc été choisi d'encapsuler la simulation de neutronique dans une interface `UnSteadyProblem`. Cette encapsulation est réalisée par le développement de la classe `Neutronic` (accessible à travers le menu Problem→SpecificProblem→Neutronic du composant TECHENTITY intégré dans SALOME).

La construction d'une instance de `Neutronic` dans l'interface graphique de SALOME présentée sur la figure 10.1, permet de visualiser (menu popup clic droit) les différentes méthodes proposées par l'interface `UnSteadyProblem` et d'autres méthodes spécifiques à la classe. L'objet séquence de calculs qui exploitera la simulation de neutronique ne considère que les méthodes de l'interface `UnSteadyProblem`.

Outre les méthodes, l'interface `UnSteadyProblem` référence les données en entrée et en sortie (calculées) de la simulation. Lors de l'appel de la méthode d'initialisation (`initialize`) les données d'entrée sont demandées à la séquence de calculs (si une séquence de calculs a été créée) par un appel de la méthode `getResults` de la séquence de calculs. Les données de sortie ne sont disponibles en lecture (méthode `getResults`) qu'après un appel réussi à la méthode `validateTimeStep`. Les méthodes `initTimeStep` et `computeTimeStep` sont appelées pour résoudre chacun des pas de temps.

Pour permettre la synchronisation des calculs, il est nécessaire de mettre en œuvre des moyens techniques permettant de bloquer l'éxécution des codes de calcul. Ainsi, sur la figure 10.2, après l'exécution de la méthode `intialize`, les adaptations du schéma de neutronique permettent de placer le code APOLLO2 dans un état d'attente de la méthode `initTimeStep` qui initialisera un pas de temps pour un prochain calcul.

[1] des calculs dépendant du temps

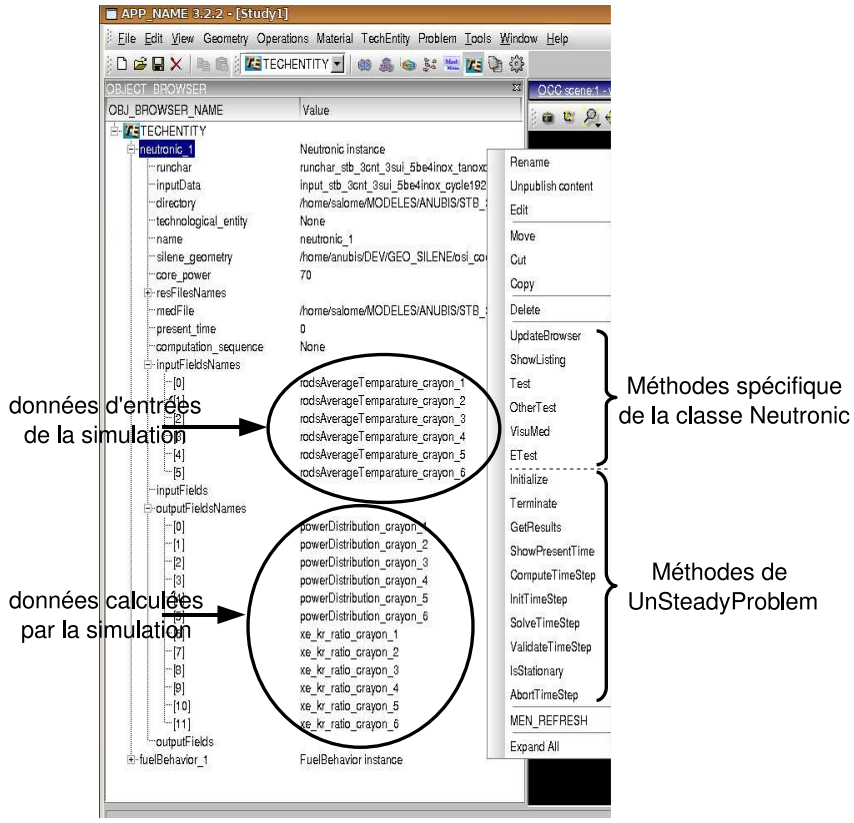


FIG. 10.1 – Présentation d'une instance de Neutronic dans l'interface SALOME

10.2 Encapsulation du comportement combustible

De la même manière que pour la neutronique, les simulations du comportement combustible des crayons REP de TANOXOS sont dépendantes du temps. Il ne s'agit donc pas d'études de problèmes stationnaires. L'application ALCYONE traitant l'étude d'un crayon REP a alors été encapsulée dans une interface `UnSteadyProblem`. Cette encapsulation est réalisée par le développement d'une classe `FuelBehavior` (accessible à travers le menu Problem→SpecificProblem→FuelBehavior du composant TECHENTITY intégré dans SALOME). Une instance de cette classe est visualisable sur la figure 10.1.

10.3 Séquencement

Après avoir construit une instance de `Neutronic`, nommée `neutronic`, et six instances de `FuelBehavior`, nommée `crayons_1` à `crayon_6`, une séquence de calcul est créée (`computationSequence_1`) et référence les sept simulations. Comme il l'a été présenté au paragraphe 7.1.5, une séquence de calculs implémente l'interface `SteadyProblem` (voir le diagramme UML de la figure 7.5) et de ce fait les méthodes d'utilisation d'une séquence de calcul sont les mêmes que celle d'une simulation.

L'exécution de la simulation couplée (figure 10.3) se résume aux simples appels successifs des méthodes d'initialisation (`initialize`), de résolution (`solve`) et de terminaison

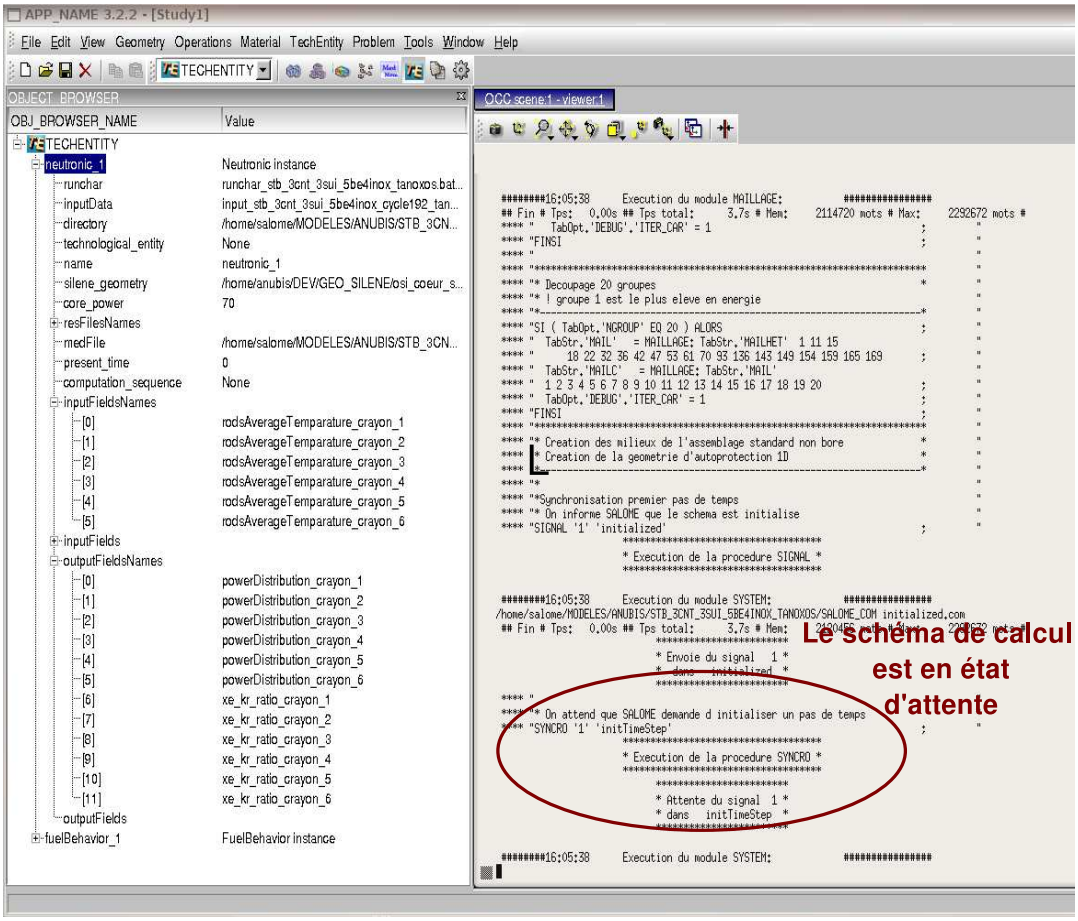


FIG. 10.2 – Lancement de la méthode Initialize d'une instance Neutronic et mise en attente du code APOLLO2

de la simulation (`terminate`).

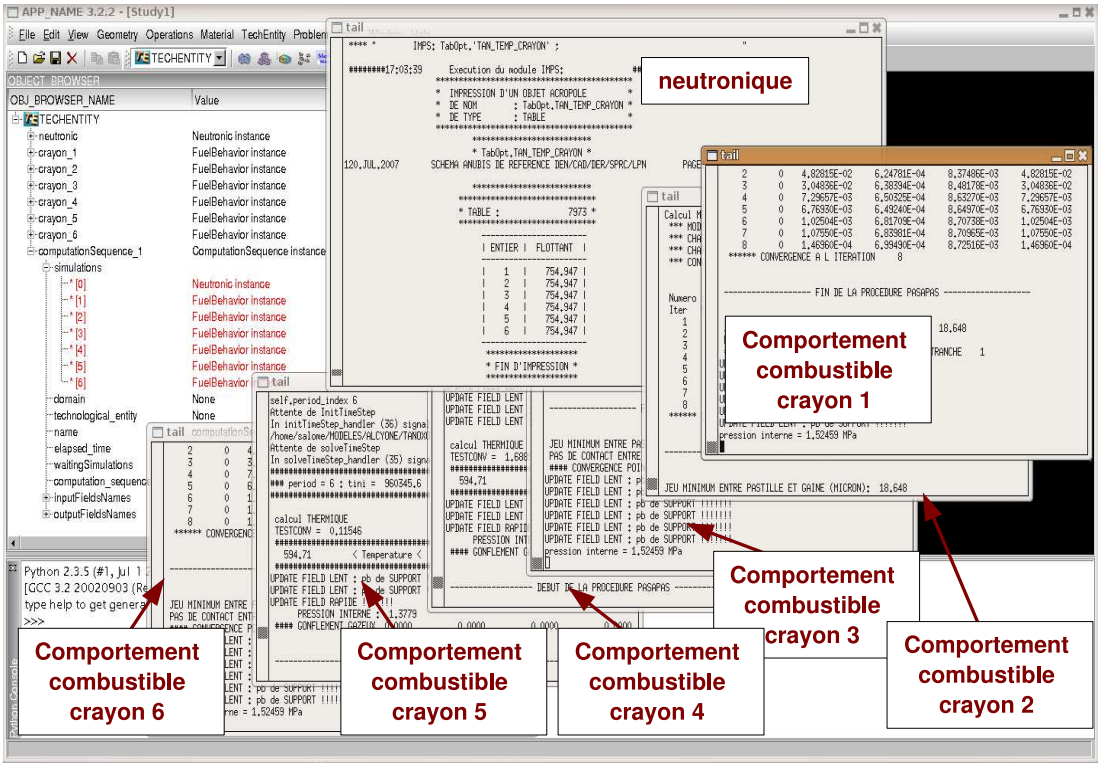


FIG. 10.3 – Exécution d'une séquence de calculs comprenant une simulation neutronique couplées avec six simulations de comportement combustible de crayons REP (crayons du dispositif TANOXOS).

Conclusion et perspectives

La problématique de départ consistait à modéliser finement des expériences d'irradiation dans des réacteurs d'irradiation technologique. Le sens accordé à *finement* s'est rapidement approché de *multiphysique*, avec des notions de couplages de simulations pour prendre en compte l'interdépendance des phénomènes physiques. Au fur et à mesure des réflexions sur les simulations de dispositifs expérimentaux, la complexité de mise en œuvre de telles modélisations a orienté le travail de thèse vers une volonté de généraliser de telles études. Cette généralisation fait alors intervenir des notions d'informatique avec des systèmes de CAO, de standardisation d'échanges de données, voire de PLM[2]. L'aspect *multiphysique* de la thèse s'étend alors au terme *pluridisciplinaire*.

Ce travail, associant des chercheurs de laboratoires d'informatique et de traitements numériques (LSIS I&M UMR CNRS 6168), ainsi que de neutronique (CEA DEN/CAD DER/SPRC/LPN), comprend à la fois une étude informatique et le commencement d'une étude physique. L'aspect informatique consistant à la définition du modèle d'Entités Technologiques, compatible avec les différents outils de simulations utilisés par les physiciens, a donné lieu à une publication lors des journées du groupe de travail en modélisation géométrique en 2006 [BDS06]. Les développements et la description des méthodes d'échanges de données entre les simulations ont quant à eux été présentés lors de la conférence internationale PLM en 2007 [BDS07]. La thèse s'est déroulée dans un laboratoire de neutronique, et ainsi, l'aspect physique a été d'autant plus approfondi dans cette discipline. Aussi, l'étude physique du travail de thèse, notamment neutronique, a été présentée aux conférences internationales PHYSOR en 2006 [BSA$^+$06] et PHYTRA en 2007 [SBd$^+$07].

Aujourd'hui, les simulations multiphysiques que nous proposons d'améliorer dans le cadre de la thèse, sont réalisées en partie de manières séquentielles et non automatiques par les physiciens. Dans le domaine de la simulation de physique des réacteurs nucléaires, les outils de CAO et de PLM sont encore peu utilisés comparés à d'autres domaines rencontrés par exemple dans l'industrie automobile. L'avènement de la plate-forme SALOME permettra dans les années à venir de franchir le pas vers la généralisation de solutions basées sur des outils de CAO, des formats de données communs et du partage de compétences des métiers du nucléaire. Une des innovations de ce travail de thèse est certainement la volonté d'intégrer les développements au sein de la plate-forme SALOME.

Le composant TECHENTITY, développé dans SALOME, est une maquette fonctionnelle, définie spécifiquement pour la modélisation de dispositifs expérimentaux. Toute-

[2] PLM : Product Lifecycle Managment

fois, le niveau de généralité obtenu permet certainement de modéliser tout autre problème de physique appliqué à tout autres domaines d'étude. Ce composant propose principalement deux fonctionnalités complémentaires pour la modélisation multiphysique de dispositifs expérimentaux.

L'une de ces fonctionnalités permet de décrire des dispositifs expérimentaux à l'aide des Entités Technologiques et d'associer un ensemble d'informations à la géométrie comme des matériaux et des résultats de simulation. A ceci s'ajoute un mécanisme de gestion de versions qui permet de définir un même objet de différentes manières, selon les besoins spécifiques des utilisateurs. Un modèle d'Entités Technologiques peut alors être vu comme une base de données organisées de manière à décrire un dispositif expérimental.

L'autre fonctionnalité du composant TECHENTITY permet de réaliser un séquencement de disciplines de physique. Il y est proposé que les différentes disciplines de physique soient contrôlables à travers des interfaces clairement définies (voir figure 7.4). A partir de cela, les outils développés sont en mesure de contrôler les échanges de données et le séquencement automatique des simulations. Il existe alors deux types d'échanges de données : ceux présents entre chaque discipline de physique et le modèle d'Entités Technologiques du dispositif expérimental modélisé, et ceux présents entre les différentes disciplines à travers une unique méthode. Cette méthode, nommée `getResults`, permet de s'affranchir des différents formalismes de stockage des résultats de chaque discipline, tels que les différents maillages de calculs rencontrés au chapitre 2.

L'implémentation de cette méthode est toutefois laissée à la charge des physiciens. Cependant, comme pour le développement des Entités Technologiques, un effort d'abstraction des difficultés informatiques rend transparent les mécanismes mis en œuvre pendant la thèse, tels que les constructions d'objets CAO, la gestion des versions, les mises en état de pause des simulations pour la synchronisation du séquencement des calculs. Ainsi, les physiciens amenés à développer ces méthodes d'adaptation se trouvent plus proches de leurs métiers ; d'autant plus que dans la majeure partie des cas, les outils de simulation sont capables de réaliser les calculs nécessaires à la mise en forme des résultats spécifiques à la méthode `getResults`.

L'exercice d'application réalisé dans la troisième partie permet de valider les avantages et les points à améliorer dans la plate-forme que nous proposons. De même, nous avons pu constater que l'adaptation de simulations existantes aux interfaces `Problem` est rapide à mettre en œuvre. A titre indicatif, l'application ALCYONE traitant du comportement combustible est développée en partie en langage PYTHON. Cette simulation a été adaptée en deux semaines environ. Le schéma de neutronique, développé en GIBIANE, est plus délicat à traiter. Cela est dû en partie à des besoins de conversions de formats d'entrée et de sortie spécifiques aux codes de calcul utilisés en neutronique[3], et aux solutions techniques qu'il a fallu mettre en œuvre pour contrôler l'éxécution du code de calcul APOLLO2. Cependant, une fois ces difficultés d'ordre technique surmontées, le schéma de neutronique a été adapté en un mois. Une des valeurs ajoutées de la thèse est certainement la rapide faisabilité de telles simulations. Auparavant, les enchaînements de tels calculs étaient très délicats à réaliser et en tout état de cause nettement plus

[3] Mais cette difficulté sera certainement levée dans les années à venir par les standard d'échanges de données proposés par le projet SALOME.

long à mettre en œuvre.

De plus, la souplesse d'utilisation des schémas de calcul, une fois adaptés aux interfaces `Problem`, facilite l'insertion d'autres simulations pour modéliser de plus en plus finement les expériences d'irradiation. Aussi a-t-il été prévu d'insérer une simulation traitant de la thermohydraulique dans l'exercice d'application. Un travail a d'ailleurs permis de constater la faisabilité quant à l'intégration d'une simulation de la thermohydraulique réalisée par le code CFD[4] TRIO_U, en méthode Volumes Eléments Finis à 3D.

Les premiers résultats des simulations de l'exercice d'application au dispositif TANOXOS correspondent bien aux grandeurs espérées en terme de température dans les crayons combustibles. A court terme, les étapes de validation et de qualification d'un cycle complet de calculs, comme d'interprétation des résultats sont une perspective de ce travail qu'il reste à effectuer dans une étude complète, où l'insertion de la thermohydraulique pourrait être ajoutée. De même, de telles études peuvent être poursuivies sur des expériences plus *dynamiques*, telles que des rampes de puissance.

A plus long terme, les capacités de modélisation des Entités Technologiques permettent d'autres utilisations qui n'ont pas été traitées dans la troisième partie. Des exercices de modélisation d'expériences et des études de plus en plus complexes permettraient de valider et d'exploiter au mieux ces fonctionnalités. Ainsi, le mécanisme de gestion de versions pourrait s'utiliser de différentes manières :

- Soit il est considéré que différentes versions d'un même modèle correspondent à différentes évolutions du dispositif expérimental. Cette utilisation permet par exemple d'étudier un dispositif tout au long de son cycle de vie, et de constater différentes adaptations (ajout d'un capteur, changement d'une pièce, ...) qui ont pu être réalisées sur celui-ci.
- Soit le mécanisme de gestion de versions permet de disposer de différentes représentations du même dispositif. Par exemple, pour modéliser un cœur de réacteur nucléaire avec des Entités Technologiques, chaque assemblage peut être décrit de manière très fine dans une version, et de manière homogène (donc plus grossière) dans une version simplifiée. Ainsi, il est envisageable d'utiliser les Entités Technologiques pour réaliser des simulations couplées multi-échelles.

La représentation orientée *Features* de notre modèle de données permet actuellement d'associer des géométries, des matériaux, des simulations et leurs résultats au travers d'atomes de modélisation : les Entités Technologiques. Il est envisageable d'enrichir ce modèle. D'une part en développant de nouveaux objets géométriques prédéfinis qui peuvent être plus complexes (opérations) et plus spécifiques (vis, écrous, grilles, ...). D'autre part, de nouvelles caractéristiques attribuant de la sémantique à ces objets pourraient être définies soit directement en tant qu'attributs des Entités Technologiques, soit en complétant la définition des matériaux des Entités Technologiques. Ces évolutions du modèle de données vont dans le sens des développements réalisés et permettraient d'aller de plus en plus loin dans la définition des objets réels, tout en faisant abstraction des techniques de simulation.

[4] Computational Fluid Dynamics

Les fonctionnalités de séquencement de disciplines de physique peuvent être utilisées pour d'autres types de problèmes. En pratique, s'il existe une séquence de calculs composée de plusieurs simulations, il est facile d'échanger l'une d'entre elles par une toute autre, calculant les mêmes champs de résultats. Dans la mesure où plusieurs simulations d'un même phénomène ont été adaptées à une interface `Problem`, il est aussi facile de les échanger et de comparer leur impact sur les résultats d'un couplage complexe. L'élaboration de telles méthodes pourraient voir le jour et apporter des compléments d'informations quant à l'étude des incertitudes sur des résultats de calculs.

Les développements réalisés ont montré la faisabilité de l'approche et son efficacité pour modéliser les expériences d'irradiation dans les réacteurs d'irradiation technologique. Notre approche pourrait facilement se généraliser à d'autres applications.

Bibliographie

[ACI00] Acis technical overview. Technical report, Spatial Technology, 2000.

[Ada04] E Adam. Xdata. Note technique, CEA DEN/DM2S/SFME/LGLS, 2004.

[AdS+04] A. Aggery, C. d'Aletto, J. Di Salvo, R. Sanchez, S. Santandrea, M. Soldevila, and G. Willermoz. The characteristics method applied to a *MTR* whole core modeling. In *PHYSOR 2004 - The physics of Fuel Cycles and Advanced Nuclear Systems : Global Developements, on CD-ROM*, Chicago, Illinois, april 25-29 2004.

[BDS06] T. Bonaccorsi, M. Daniel, and J. Di Salvo. Modèles géométriques et physiques pour une plate-forme de simulations numériques d'expériences d'irradiations technologiques. In *actes des journées GTMG 2006*, ENS Cachan, France, 10-14 mars 2006.

[BDS07] T. Bonaccorsi, M. Daniel, and J. Di Salvo. Data exchange interface and model for coupling softwares in nuclear reactor simulations. In *International conference in Product Lifecycle Management*, pages 207–216, Milano, Italy, july 2007.

[BMN94] J. Both, B. Morillon, and J. Nimal. A survey of tripoli-4. In *8th International Conference on Radiation Shielding*, pages 373–380, Arlington, Etats-unis, april 24-27 1994.

[Bon05] T. Bonaccorsi. Evaluation des objets technologiques de la plate-forme de neutronique descartes dans le cadre de la modélisation multiphysique des dispositifs expérimentaux d'osiris et du rjh. Note technique, CEA DEN/DER/SPRC/LPN, 2005.

[Bou95] M. Bouazza. *La norme STEP.* Hermes, Paris, France, 1995.

[Bri93] J. F. Briesmeister. MCNP–A general monte carlo N-particle transport code, version 4A. Report LA-12625-M, Los Alamos National Laboratory, 1993.

[Bru05] X. Bruyet. Description du modèle commun descartes/neptune. Note technique, EDF/R&D/SINETICS, 2005.

[BSA+06] T. Bonaccorsi, J. Di Salvo, A. Aggery, C. D'aletto, C. Döderlein, P. Sireta, G. Willermoz, and M. Daniel. Development of a multi-physics calculation platform dedicated to irradiation devices in a material testing reactor. In *Physor 2006 American Nuclear Society Topical Meeting on Advances in Nuclear Analysis and Simulation, on CD-ROM*, Vancouver, Canada, september 10-14 2006.

[Cas07a] Open Cascade. Open cascade, simulation integrator. *http://www.opencascade.com/*, 2007.

[Cas07b] Open Cascade. Salome : The open source integration platform for numerical simulation. *http://www.salome-platform.org*, 2007.

[CL01] J. Corney and T. Lim. *3D Modeling with ACIS.* Saxe-Coburg publications, Glasgow, Great Britain, 2001.

[CM07] D. Cazier and C. Minich. *Informatique Graphique, modélisation géométrique et animation, Dominique Bechmann et Bernard Péroche, Les modèles classiques.* Lavoisier-Hermès, 2007.

[Dan07] M. Daniel. *Informatique Graphique, modélisation géométrique et animation, Dominique Bechmann et Bernard Péroche, Courbes et surfaces paramétriques.* Lavoisier-Hermès, 2007.

[DCD+98] Ph. Dehaudt, L. Caillot, G. Delette, G. Eminet, and A. Mocellin. *Irradiation of UO_{2+x} fuels in the TANOX device.* IAEA-TECDOC-1036, Vienna, Austria, 1998.

[EC04] EDF and CEA. Documentation de la bibliothèque med-fichier v2.2. *http://www.code-aster.org/outils/med/*, 18 novembre 2004.

[FG99] P. J. Frey and P. L. George. *Maillages. Applications aux éléments finis.* Hermes, Paris, France, 1999.

[Fic98] E. Le Fichoux. Presentation et utilisation de *CASTEM 2000. http ://www-cast3m.cea.fr*, 1998.

[GAM98] Groupe GAMA. *Conception de produits mécaniques, méthodes, modèles et outils, Michel Tollenaere, Modélisation par entités.* Hermès, 1998.

[Gar03] Y. Gardan. Cao : Vers la modélisation fonctionnelle. In *Techniques de l'ingénieur - Mathématiques pour l'ingénieur, H3752*, 2003.

[GF03] M. Grandotto and P. Fillion. NEPTUNE TechObj component v0.4-b1. technological objects user's guide and reference manual. Note technique, CEA DEN/DTN DEN/DM2S, 2003.

[Gir04] P. Giroux. Langage UML : développement de logiciel et modélisation visuelle. In *Techniques de l'ingénieur, H3238*, 2004.

[Gra04] M. Grandotto. Neptune component interpolation, reference manual and users guide. Note technique, CEA DEN DTN/SMTM/LMTR, 2004.

[GT96] Th. De Gramont and I. Toumi. Isas reference manual. Note technique, CEA DEN DM2S/SERMA/LETR, 1996.

[HAB+05] N. Huot, A. Aggery, D. Blanchet, C. D'Aletto, J. Di Salvo, C. Döderlein, P. Sireta, and G. Willermoz. The JHR neutronics calculation scheme based on characteristics method. In *Mathematics and Computation, Supercomputing, Recator Physics and Nuclear ans Biological Applications, on CD-ROM*, Avignon, France, sept 12-15 2005.

[Hof89] C. M. Hoffmann. *Geometric and Solid Modeling : An Introduction.* Morgan Kaufmann, 1989.

[HVP05] P. Heyraud, V.Marelle, and D. Plancq. Maquette du modèle de donées utilisateur de PLEIADES (ALCYONE). Note technique, CEA DEN/CAD DEC/SESC/SLC, 2005.

[IGE] The iges 5.x preservation society homepage. *http://www.iges5x.org/*.

[JMS07] C. Jermann, D. Michelucci, and P Schreck. *Informatique Graphique, modélisation géométrique et animation, Dominique Bechmann et Bernard Péroche, Modélisation géométrique par contraintes.* Lavoisier-Hermès, 2007.

[KSG00] A. Kumbaro, V. Seignole, and J.M. Ghidaglia. Flux schemes for the two-fluid models of the *TRIO_U* code. In *AMIF-ESF workshop,* Computing Methods for Two-Phase Flow, Aussois, France, january 12-14 2000.

[Lab07] U.S. Army Ballistic Research Laboratory. Brl-cad. *http://www.brlcad.org/*, 2007.

[Lee05] S. H. Lee. A CAD-CAE integration approach using feature-based multi-resolution and multi-abstraction modelling techniques. In *Computer-Aided Design*, volume 37, pages 941–955, 2005.

[Lem04] C. Lemaignan. *Science des matériaux pour le nucléaire*. EDP SCIENCES, Collection Génie Atomique, Les Ulis, France, 2004.

[Li01] C. L. Li. A feature-based approach to injection mould cooling system design. In *Computer-Aided Design*, volume 33, pages 1073–1090, 2001.

[LLFM90] J.J. Lautard, S. Loubière, and C. Fedon-Magnaud. CRONOS, a modular computational system for neutronic core calculations. In *IAEA topical meeting, Advanced calculational methods for power reactors*, Cadarache, France, 1990.

[LMJL04] V. Lefebvre, O. Morvant, C. De Saint Jean, and J. J. Lautard. Projet descartes : Définition du modèle de données interne. Note technique, CEA DEN DTN/SMTM/LMTR, 2004.

[LSC+99] S. Loubière, R. Sanchez, M. Coste, A. Hébert, Z. Stankovski, C. Van Der Gucht, and I. Zmijarevic. Apollo2 *Twelve Years Later*. In *International Conference on Mathematics and Computation, Reactor Physics and Environmental Analysis in Nuclear Applications*, volume 2, pages 1298–1315, Madrid, Espagne, September 27-30 1999.

[MD04] R. Maculet and M. Daniel. Conception, modélisation géométrique et contraintes en CAO : Une synthèse. In *Revue d'Intelligence Artificielle*, volume 18, pages 619–645, 2004.

[MRH94] G. Marleau, R. Roy, and A. Hébert. DRAGON : A collision probability transport code for cell and supercell calculations. Report IGE-157, Institut de génie nucléaire, École Polytechnique de Montréal, Montréal, Québec, 1994.

[MY98] M. Mekhilef and B. Yannou. Conception intégrée assistée par ordinateur. In *Techniques de l'ingénieur, BM5006*, 1998.

[nur07] Nuresim project : Nuclear reactor simulations. *http://www.nuresim.com*, 2007.

[PB05] M. PAOLILLO and X. Bruyet. Description technique du prototype modèle de données commun descartes/neptune/pléiades, version v0.3.1. Note technique, EDF/R&D/SINETICS, 2005.

[Per06] F. Perdu. Proposition d'une interface de couplage en vue des couplages de codes dans neptune version 1. Note technique, CEA DEN/DER/SSTH/LMDL, 2006.

[Pie99] G. Pierra. Représentation et échange de données techniques. In *Actes des Entretiens POLYMECA, pp 35-54*, Valenciennes, France, 24 novembre 1999.

[POV07] POV-Ray the persistence of vision raytracer. *http://www.povray.org/*, 2007.

[Req80] A. A. G. Requicha. Representations for rigid solids : Theory, methods, and systems. In *ACM Computing Surveys*, volume 12, pages 437–464, 1980.

[Reu03] P. Reuss. *Précis de neutronique*. EDP sciences, collection génie atomique, 2003.

[Roy98] R. Roy. The cyclic characteristics method. In *International Conference on the Physics of Nuclear Science and Technology*, volume 1, page 407, Long Island, New York, October 5-8 1998.

[Sal02] J. Di Salvo. *Contribution à l'étude des incertitudes des paramètres neutroniques d'un cœur compact et hétérogène : le réacteur d'irradiation Jules Horowitz.* PhD thesis, Université Louis Pasteur de Strasbourg, 2002.

[SBAP04] N. Schmidt, L. Buffe, M. Auclair, and B. Pouchin. Thermal assessment for the design of a tensile test device for the Jules Horowitz Reactor(JHR). Note technique, CEA DEN/CAD DER/SESI, CEA DEN/SAC DRSN/SIREN, 2004.

[SBd+07] J. Di Salvo, T. Bonaccorsi, C. d'Aletto, C. Döderlein, O. Gueton, and P. Sireta. The osiris reactor neutronic reference calculation scheme based on the method of characteristics. In *PHYTRA-1 : The First International Conference on Physics and Technology of Reactors and Applications, on CD-ROM*, Marrakech, Morocco, march 14-16 2007.

[SCCM03] S.Giurgea, J.L. Coulomb, T. Chevalier, and Y. Marechal. Salome une plateforme générique de simulation multiphysique : Exemple d'application électrothermique. In *NUMELEC 2003 : 4th European Conference on Numerical Methods in Electromagnetism, on CD-ROM*, Toulouse, France, octobre 28-30 2003.

[SG05] S. Subramani and B. Gurumoorthy. Maintaining associativity between form feature models. In *Computer-Aided Design*, volume 37, pages 1319–1334, 2005.

[Spi02a] P. Spiteri. Introduction à la méthode des éléments finis. In *Techniques de l'ingénieur - Mathématiques pour l'ingénieur, AF504*, 2002.

[Spi02b] P. Spiteri. Méthode des différences finies pour les *EDP* d'évolution. In *Techniques de l'ingénieur - Mathématiques pour l'ingénieur, AF501*, 2002.

[Spi02c] P. Spiteri. Méthode des différences finies pour les *EDP* stationnaire. In *Techniques de l'ingénieur - Mathématiques pour l'ingénieur, AF500*, 2002.

[Spi02d] P. Spiteri. Présentation générale de la méthode des éléments finis. In *Techniques de l'ingénieur - Mathématiques pour l'ingénieur, AF505*, 2002.

[Spi03] P. Spiteri. Approche variationnelle pour la méthode des élements finis. In *Techniques de l'ingénieur - Technologies logicielles Architectures des systèmes, AF503*, 2003.

[SS93] R. Sanchez and Z. Stankovski. Silene & tdt : A code for collision probability calculations in xy geometries. In *ANS Annual Meeting*, pages 458–460, San Diego, California, June 20-24 1993.

[Sta96] Z. Stankovski. Silene : A graphical user interface for 2d/3d pre & post processing. In *Seminar on 3D Deterministic Radiation Transport Codes, on CD-ROM*, Paris, France, December 2-3 1996.

[Sta97] Z. Stankovski. La java de silène : A graphical user interface for 3d pre & post processing. In *Joint International Conference on Mathematical Methods and Supercomputing for Nuclear Applications*, Saratoga Springs, NY USA, October 6-10 1997.

[swi] SWIG :Simplified Wrapper and Interface Generator. *http://www.swig.org*.

[Taj03] M. Tajchman. Intégration de composants dans l'environnement PAL/salomé. Note technique, CEA DEN/SAC DM2S/SFME/LGLS, 2003.

[TGR97] I. Toumi, D. Gallo, and E. Royer. Advanced numerical methods for three dimensional two-phase flow calculations in *PWR*. In *NURETH-8*, volume 3, page 1684, Kyoto, Japan, september 30 - october 4 1997.

[vR96] G. van Rossum. *Python Reference Manual*. Stichting Mathematisch Centrum, Amsterdam, 1996.

[WAB+04] G. Willermoz, A. Aggery, D. Blanchet, C. Chicoux, J. Di Salvo, C. Doderlein, D. Gallo, F. Gaudier, N. Huot, S. Loubière, and B. Noël. Horus3d code package development and validation for the jhr modeling. In *Physor 2004, The Physics of Fuel Cycles and Advanced Nuclear Systems : Global Developments, on CD-ROM*, Chicago, Illinois, april 25-29 2004.

[Zie] James Ziegler. The stopping and range of ions in matter. *http://www.srim.org/*.

[Zie73] O.C. Zienkiewicz. *La méthode des éléments finis appliquée à l'art de l'ingénieur*. EDISCIENCE, Paris, 1973.

Annexe A

Etude d'un crayon REP sous irradiation

Les paragraphes suivants présentent l'articulation générale d'un couplage résolu de manière analytique sur l'exemple d'un crayon combustible aux conditions de fonctionnement d'un Réacteur à Eau Pressurisée.

A.1 Transferts thermiques dans le combustible

L'objectif de la résolution de l'équation de la thermique est de fournir d'une part la température moyenne du combustible et d'autre part la température au centre des crayons. La température moyenne est ensuite utilisée comme point d'entrée dans les calculs de contre-réaction neutroniques. Dans le crayon combustible, le principal mode de transmission de la chaleur est la conduction, au sein de la pastille, dans le jeu pastille/gaine et dans la gaine. La conduction dans un solide suit la loi de Fourier :

$$\overrightarrow{\phi''} = -\lambda(T)\overrightarrow{gradT}$$

où λ désigne la conductivité thermique qui vaut approximativement 4.4 $Wm^{-1}K^{-1}$ à 500 ° C pour de l'UO_2 et ϕ'' représente le flux de chaleur moyen obtenu par le calcul de neutronique (environ 60 W/cm^2 dans un REP). En régime permanent, la production ϕ''' est égale à la dissipation $div\overrightarrow{\phi''}$. En coordonnées cylindriques, il faut donc résoudre l'équation suivante :

$$-\int_{T(0)=T_{max}}^{T(r)} \lambda(T)dT = \frac{\phi''' r^2}{4}$$

Pour une pastille cylindrique pleine, avec une conductivité thermique constante, le gradient maximal de température entre le centre de la pastille et la périphérie est donné par :

$$\Delta T_{max} = \frac{\phi'}{4\pi\lambda}$$

où ϕ' désigne la puissance linéique moyenne du crayon de l'ordre de 200 W/cm dans un REP. Avec les valeurs précédentes, le gradient de température est donc de l'ordre de 360 ° C.

En négligeant le terme source dans la gaine et dans le jeu pastille/gaine (92 % de la puissance thermique du réacteur provient de l'énergie de fission générée dans la pastille, les 8 % restants sont générés directement dans le modérateur et les matériaux de structure), la production de chaleur est égale au flux provenant de la pastille. Le jeu est d'épaisseur trop faible pour qu'un mouvement de convection naturelle s'établisse (jeu d'hélium d'épaisseur $e = 50\mu m$, présentant une meilleure conduction que l'air ; $\lambda_{jeu}(400\,^\circ\mathrm{C}) = 0.27 W m^{-1} K^{-1}$). En coordonnées cylindriques, il faut donc résoudre l'équation suivante :

$$\frac{\phi'}{2\pi r} = -\lambda(T)\frac{dT}{dr}$$

Le gradient de température sur le jeu et sur la gaine est donné par :

$$\Delta T_{jeu} = \frac{\phi'' e}{\lambda_{jeu}}; \Delta T_{gaine} = \frac{\phi'' E}{\lambda_{gaine}}$$

avec $E = 500\mu m$ et $\lambda_{gaine}(400\,^\circ\mathrm{C}) = 17.4 W m^{-1} K^{-1}$ d'où ΔT_{jeu} =110 $^\circ$ C et ΔT_{gaine} =17 $^\circ$ C .

A.2 Transferts thermiques entre le crayon et le fluide

La convection forcée du fluide primaire est le mode de transfert le plus efficace. Plus la vitesse du fluide est élevée, plus le transfert thermique est élevé. Les écoulements sont donc le plus souvent turbulents et les lois physiques principalement empiriques, car les phénomènes physiques en régime turbulent avec transfert thermique ne suivent pas de modèles simples. De ce fait, il est généralement défini un coefficient de transfert thermique entre la surface d'un solide à la température T_S et un fluide à la température T_F par

$$h = \frac{\phi''}{T_S - T_F} = \frac{\phi''}{\Delta T_{mod}}$$

En écoulement simple phase, vapeur ou liquide, ou encore lorsque l'ébullition nucléée est partielle, on utilise l'équation de Dittus et Boelter :

$$Nu = 0.023 Re^{0.8} Pr^{0.4} = \frac{h D_h}{\lambda}$$

- D_h est le diamètre hydraulique représentant la surface de passage du liquide ;
- Re est le nombre de Reynolds (voir paragraphe 2.3) ;
- $Pr = \mu c_p/\lambda$ est le nombre de Prandtl. Il donne l'impact des propriétés thermodynamiques du fluide sur le transfert thermique ;
- Nu est nombre de Nusselt. Il indique le rapport de la quantité de chaleur échangée par convection à celle échangée par conduction ;

A 155 bars et environ 300 $^\circ$ C, les grandeurs thermodynamiques de l'eau liquide sont les suivantes :

- Densité : $\rho = 727 kg m^{-3}$
- Conductivité thermique : $\lambda = 0.56 W m^{-1} K^{-1}$
- Viscosité dynamique : $\mu = 9^{-5}$ Poiseuille (Pa.s)
- Capacité thermique à pression constante : $c_p = 5500 J kg^{-1} K^{-1}$

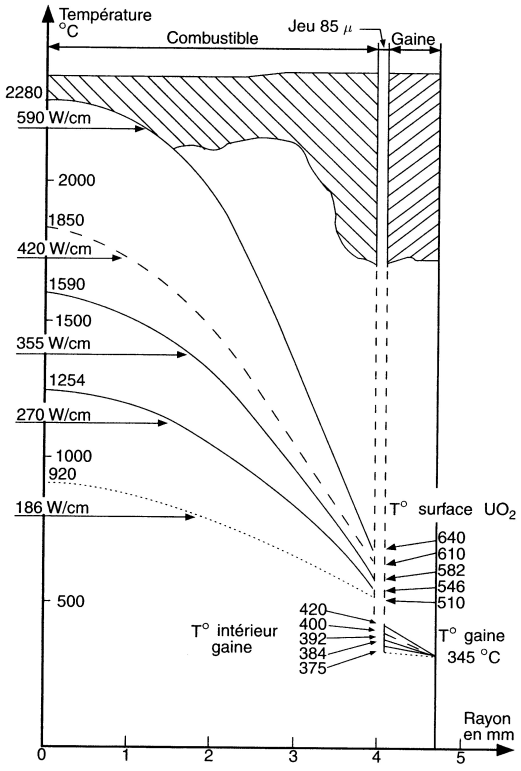


FIG. A.1 – Variation de la température dans un crayon combustible en fonction de la puissance linéique

Dans les conditions REP, les valeurs numériques sont donc les suivantes :

- $U = 5ms^{-1}$
- $D_h = 1.1^{-2}m$
- $Re = 444300$
- $Pr = 0.88$
- $Nu = 720$

D'où $h = 36680Wm^{-2}K^{-1}$ et ΔT_{mod} =16 $^\circ$ C .

La figure A.1 présente différents profils de températures rencontrés selon la puissance linéique du réacteur. C'est ce profil de température qui a un impact, en neutronique, sur les sections efficaces.

A.3 Température introduite dans les calculs neutroniques

Les variations de température dans un réacteur nucléaire entraînent des variations des sections efficaces qu'il convient de prendre en compte. La répartition radiale de la température combustible dans la pastille et sa variation au cours de l'irradiation peuvent être obtenues par un calcul de thermomécanique (code METEOR ou application ALCYONE du code PLEIADES). Le formalisme d'autoprotection recommandé dans les schémas de référence pour les REP ne peuvent pas prendre en compte le fort gradient de température (300 $^\circ$ C) présent dans le crayon combustible. Dans ce cas, il est défini une

température effective, calculée de manière à conserver le taux de réaction d'absorption de l'^{238}U. L'expression de la température effective qu'il est recommandé d'utiliser pour les calculs APOLLO2 est la suivante :

$$T_{eff} = \overline{T_{therm}} - \frac{1}{18}\left(T_c - T_s\right)$$

où $\overline{T_{therm}}$ est la température moyenne thermique :

$$\overline{T_{therm}} = \frac{\int_0^R T\left(\vec{r}\right) d\vec{r}}{\int_0^R d\vec{r}}$$

En réalité, du fait de la combustion massique et de la forte puissance sur le bord de la pastille, le profil de température n'est pas tout à fait parabolique. Il est également appliqué une correction liée à la prise en compte des liaisons cristallines, qui est d'autant plus faible que la température dans le combustible est élevée. L'élargissement des résonances est tabulé dans APOLLO2 sous la forme de sections efficaces effectives en fonction de la température. L'élargissement Doppler (augmentation de la capture neutronique avec la température) est réalisé dans un modèle de Gaz Libre, alors que les atomes du combustible n'ont pas un spectre d'agitation thermique maxwellien, du fait qu'ils sont insérés dans un système cristallin (atomes de ^{238}U dans l'oxyde UO_2). Pour prendre en compte cet effet, la température effective recommandée pour le traitement APOLLO2 de l'effet Doppler des combustibles UO_2 est donnée par la formule suivante :

$$\frac{T_{eff}}{T} = 1 + \frac{8.6}{T} + \frac{3100}{T^2}$$

avec T la température en Kelvin.

Cette correction de la température thermique pour prendre en compte les effets de liaisons cristallines est donc assez faible : elle varie de $+18.9\,^{\circ}$C à température ambiante (Tcomb $= 27\,^{\circ}$C) à $+12\,^{\circ}$C à température nominale (Tcomb $= 650\,^{\circ}$C).

A.4 Effets de contre-réaction

Ces nouvelles températures sont donc introduites dans le calcul neutronique pour obtenir un nouveau calcul de la puissance linéique dissipée au sein de la pastille, qui servira de donnée d'entrée pour les nouveaux calculs de thermique. Cette variation des températures joue un double rôle sur le plan neutronique :

- Un premier effet extrêmement rapide est de modifier les valeurs des sections efficaces neutronique résonnantes, conduisant à une augmentation de l'absorption résonnante des noyaux lourds du combustible lorsque la température augmente (effet Doppler lié à l'agitation thermique des atomes que vont rencontrer les neutrons) ;
- La variation de la température du modérateur entraîne une variation de sa densité ainsi que de la répartition en énergie des neutrons, au travers des changements des sections efficaces (absorption, diffusion et transferts d'énergie) liés au phénomène de dilatation thermique du modérateur. Les coefficients associés sont très variables selon les matériaux et selon les filières de réacteurs.

Annexe B

Notions de section efficace

En neutronique, la section efficace d'une interaction de type i (i représentant les intéractions de diffusion, de capture, de fission, ...) représente la probabilité qu'il y ait une interaction i lors de la collision d'un neutron avec un noyau. On distingue les sections efficaces dites *microscopiques*, notées σ_i, et les section efficaces dites *macroscopiques*, notées Σ_i.

Une section efficace microscopique est caractéristique d'une cible individuelle, c'est-à-dire du noyau d'un isotope donné. Ainsi, il existe des sections efficaces microscopiques σ_i différentes pour tous les isotopes. L'ordre de grandeur de σ_i est de $10^{-24}cm^2$, les physiciens expriment les sections efficaces microscopiques en *barn*, avec $1barn(b) = 10^{-24}cm^2$.

Une section efficace macroscopique est caractéristique d'un matériau contenant un grand nombre de cibles. Elle ramène la section efficace microscopique à un volume de densité de noyaux N, tel que $\Sigma_i = N\sigma_i$. Dans ce cas Σ_i s'exprime en cm^{-1}.

Lorsque le milieu est composé de k isotopes différents, de densité N_k et de sections efficaces microscopiques $\sigma_{i,k}$, alors la section efficace macroscopique du milieu est :

$$\Sigma_i = \sum_k N_k \sigma_{i,k} \tag{B.1}$$

Ainsi, la section efficace macroscopique, qui apparaît dans l'équation de Boltzmann, peut être influencée :

- par des modifications de la composition isotopique et des densités N_k en isotope k, dues à l'évolution des matériaux, aux changements d'état des matériaux et aux contraintes de pression ;
- par des variations des sections efficaces microscopiques $\sigma_{i,k}$, dues aux variations de la température (effet Doppler) et aux traitements numériques (condensation en énergie et autoprotection).

Annexe C

Introduction à la norme STEP et au protocole d'application 214

C.1 Généralité

La norme STEP (Standard for the Exchange of Product model data) ISO 10303 [Bou95] a été développée pour résoudre les problèmes d'échange, de partage et d'intégration des modèles d'informations dans des domaines industriels différents [Pie99]. Elle est décrite par un ensemble de composantes définies chacune par plusieurs documents standard. Ces composantes sont :

- la méthode de description constituée du langage EXPRESS ;
- la méthode d'implémentation constituée du format d'échange et de l'interface de manipulation des données STEP ;
- les ressources intégrées composées de ressources générales (indépendantes du contexte d'application) et de ressources spécifiques à un groupe de domaine d'application ;
- une liste de protocoles d'applications, dont le protocole AP214 (AUTOMOTIVE DESIGN) que nous avons utilisé. Ces protocoles d'application sont des cas d'utilisations spécifiques à un domaine ;
- la méthode de test de conformité des implémentations des protocoles d'application.

Ces composantes font référence à différentes parties de la norme STEP, numérotées de façon unique et non ambiguë (voir tableau C.1).

Composantes	**Parties**
Méthodes de description	1x
Ressources Intégrées - Ressources génériques	4x à 9x
Ressources intégrées - Ressource d'application	1xx
Protocole d'application	2xx à 11xx
Méthodologie des tests de conformité	3x
Suites de tests abstraits	12xx à 21xx
Méthodes d'implémentation	2x

Tab. C.1 – Numérotation des composantes de la norme STEP.

Le principal objectif de la norme STEP est de fournir un mécanisme neutre pour décrire les informations d'un produit durant les différentes étapes de son cycle de vie (conception, validation, utilisation, maintenance). La norme STEP va au-delà d'une simple description d'un format d'échange de données. Il s'agit plus particulièrement d'une méthodologie STEP qui permet d'une part de spécifier des modèles de données avec un langage formel (le langage EXPRESS) ; d'autre part, elle offre des méthodes d'implémentation et de manipulation des données d'un modèle EXPRESS, de manière indépendante de tout support matériel et logiciel.

C.2 Les protocoles d'application

Un protocole d'application est une partie de la norme STEP. Il définit un domaine, les besoins d'informations pour une application donnée, et des méthodes STEP utilisées pour exprimer ce besoin. La liste de documents associés aux protocoles d'applications est croissante dans le temps puisque lorsqu'un domaine d'activité utilise la norme STEP pour développer des solutions d'échange, de partage et d'intégration, un nouveau protocole d'application est à définir dans la norme. Lors de l'utilisation, un protocole d'application peut-être vu comme un paquetage dans lequel est défini un certain nombre de classes relatives à un domaine spécifique. Ces classes sont définies en langage EXPRESS et la plupart du temps traduites dans des langages plus exploitables par des applications informatiques , tels que C++, Fortran et JAVA.

C.3 L'interface standard d'accès aux données (SDAI)

Le format d'échange de données STEP constitue une partie de la norme (part 21 : Clear text encoding of the exchange structure). Les données à échanger sont définies dans un modèle STEP, ce modèle est décrit dans un ou plusieurs fichiers de données STEP (C.2). Les fichiers de données sont de type fichiers textes d'extension *.step* ou *.stp* Ces fichiers sont conformes à la partie 21 de la norme. Physiquement, un fichier de données STEP se compose de deux sections importantes :

Le SDAI (Standard Data Access Interface) met à disposition des opérations permettant de manipuler les instances d'une application. L'environnement du SDAI est caractérisé par la notion d'état, il est décrit par le schéma d'une session. Une session SDAI décrit les concepts tels que l'emplacement des données manipulées, leurs structures, le contrôle d'accès, les conditions d'erreurs. Le SDAI se voit comme une interface qui propose une bibliothèque de fonctionnalités dont des opérations de l'environnement pour l'initialisation et la création de sessions, les opérations d'une session pour la gestion des emplacements des données (fichiers de données), la création d'instances (lecture de fichiers de données), les opérations sur la hiérarchie des entités...

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(('Open CASCADE Model'),'2;1');
FILE_NAME('Open CASCADE Shape Model','2005-07-22T11:32:54',('Author'),(
    'Open CASCADE'),'Open CASCADE STEP processor 5.2','Open CASCADE 5.2'
  ,'Unknown');
FILE_SCHEMA(('AUTOMOTIVE_DESIGN_CC2 { 1 2 10303 214 -1 1 5 4 }'));
ENDSEC;
DATA;
#1 = APPLICATION_PROTOCOL_DEFINITION('committee draft',
  'automotive_design',1997,#2);
#2 = APPLICATION_CONTEXT(
  'core data for automotive mechanical design processes');
#3 = SHAPE_DEFINITION_REPRESENTATION(#4,#10);
#4 = PRODUCT_DEFINITION_SHAPE('','',#5);
#5 = PRODUCT_DEFINITION('design','',#6,#9);
#6 = PRODUCT_DEFINITION_FORMATION('','',#7);
#7 = PRODUCT('Product 18','Product 18','',(#8));
#8 = MECHANICAL_CONTEXT('',#2,'mechanical');
#9 = PRODUCT_DEFINITION_CONTEXT('part definition',#2,'design');
#10 = SHAPE_REPRESENTATION('',(#11,#15,#19,#23,#27,#31,#35,#39,#43,#47,
    #51,#55,#59,#63,#67,#71,#75,#79,#83,#87,#91,#95,#99,#103,#107,#111,
    #115,#119,#123,#127,#131,#135,#139,#143,#147,#151,#155,#159,#163,
    #167,#171,#175,#179,#183,#187,#191,#195,#199),#203);
#11 = AXIS2_PLACEMENT_3D('',#12,#13,#14);
#12 = CARTESIAN_POINT('',(0.,0.,0.));
#13 = DIRECTION('',(0.,0.,1.));
#14 = DIRECTION('',(1.,0.,-0.));
#15 = AXIS2_PLACEMENT_3D('',#16,#17,#18);
#16 = CARTESIAN_POINT('',(0.,0.,0.));
#17 = DIRECTION('',(0.,0.,1.));
#18 = DIRECTION('',(1.,0.,-0.));
#19 = AXIS2_PLACEMENT_3D('',#20,#21,#22);
#20 = CARTESIAN_POINT('',(0.,0.,0.));
#21 = DIRECTION('',(0.,0.,1.));
#22 = DIRECTION('',(1.,0.,-0.));
#23 = AXIS2_PLACEMENT_3D('',#24,#25,#26);
#24 = CARTESIAN_POINT('',(0.,0.,0.));
#25 = DIRECTION('',(0.,0.,1.));
#26 = DIRECTION('',(1.,0.,-0.));
...
...
...
ENDSEC;
END-ISO-10303-21;
```

TAB. C.2 – Exemple d'un fichier de données STEP ISO-10303-21.

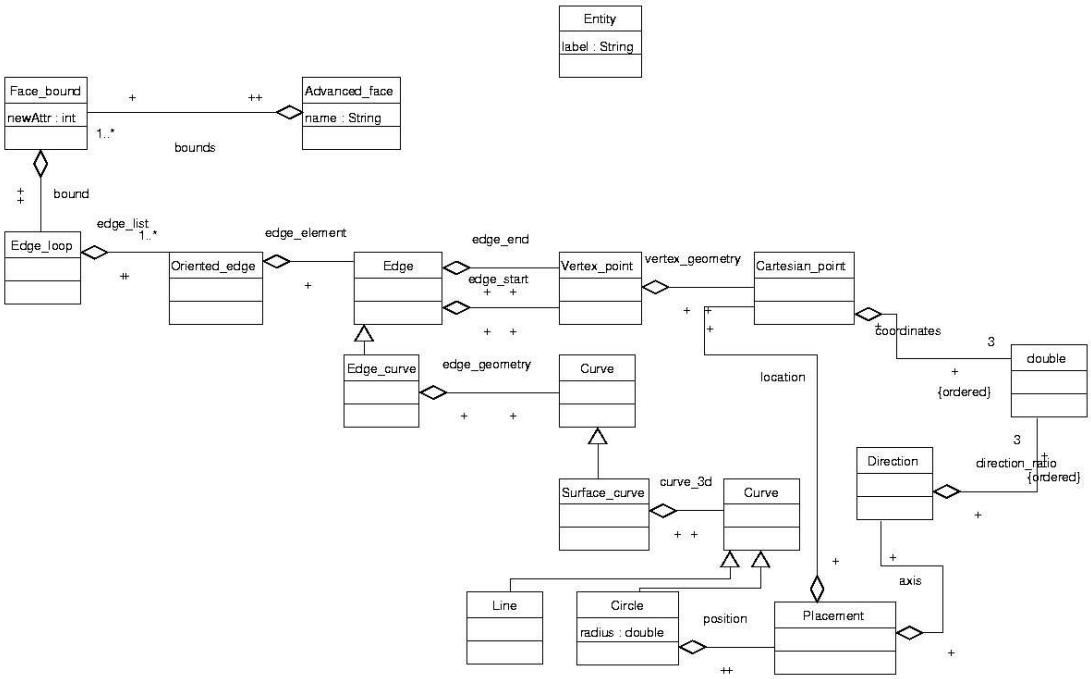


FIG. C.1 – Diagramme de classes (incomplet) du protocole d'application 214 utilisé.

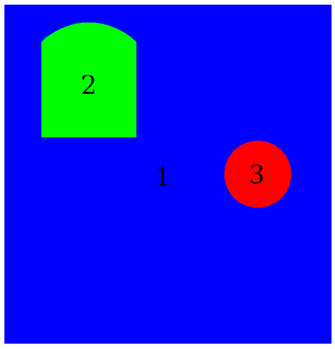


FIG. C.2 – Exemple de géométrie.

C.4 Le protocole d'application 214

Le composant GEOM de SALOME utilise le protocole d'application 214 AUTOMOTIVE DESIGN (FILE_SCHEMA tableau C.2) pour échanger les géométries. Un diagramme de classe (figure C.1) a été déduit à partir de la documentation EXPRESS. Toutes les classes dérivent de la classe Entity (Sauf les agrégations qui dérivent de la classe AEntity) qui possède l'attribut label (La classe AEntity ne possède pas cet attribut) correspondant au numéro de l'instance associée dans le fichier de données (Tableau C.2).

La figure C.2 constitue un exemple de géométrie à partir duquel nous étudierons la sémantique de chacune des classes de la figure C.1. La géométrie est en 2D plane, elle est composée de trois faces numérotées 1,2 et 3. Dans le diagramme de la figure C.1, à chaque face est associé un objet de la classe Advanced_face (figure C.5).

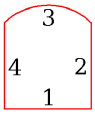


FIG. C.3 – Périmètre de la face 2.

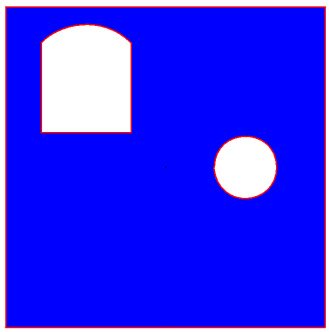


FIG. C.4 – Face 1 et ses périmètres.

Une face est limitée par son périmètre (figure C.3). Le périmètre d'une face est représenté par un objet de la classe Face_bound. A partir de l'objet Advanced_face, l'attribut bounds permet d'accéder aux périmètres (agrégation de Face_bound) de la face.

La face 1 de notre exemple est *trouée*, dans ce cas l'agrégation accessible par l'attribut bounds de l'objet Advanced_face de la face 1, possède trois instances de Face_bound (figure C.5) correspondant aux trois périmètres disjoints (figure C.4).

Le périmètre de la face 2 (figure C.3) est fermé, c'est-à-dire que l'ensemble des bords forme une boucle. L'attribut bound de la classe Face_bound fait référence à un objet de la classe Edge_loop. L'attribut edge_list de la classe Edge_loop permet d'accéder à l'agrégation contenant chacun de ses bords orientés (Oriented_edge).

Pour la face 2, l'agrégation de l'attribut edge_list contient quatre instances de la classe Oriented_edge (figure C.6) qui représente trois segments (bord 1,2 et 4 de la figure C.3) et un arc de cercle (bord 3).

Le bord 1 de la face 2 est un segment. La représentation STEP AP214 définit ce bord par un point de départ et un point d'arrivée accessibles par les attributs edge_start et edge_end de l'instance de la classe Edge_curve (figure C.7). Le type de l'attribut curve_3d de l'instance de la classe Surface_curve précise que ce bord est un segment (Line).

Comme pour les segments, les arcs de cercles possèdent un point de départ et d'arrivée. La différence avec les segments apparaît au niveau de l'attribut curve_3d (Circle) de l'instance de la classe Surface_curve. Dans le cas du bord 3 (Arc de cercle figure C.8), l'instance de la classe Circle possède un attribut radius qui renseigne sur le rayon du cercle (de l'arc de cercle). L'attribut position de l'instance de Circle nous renseigne sur

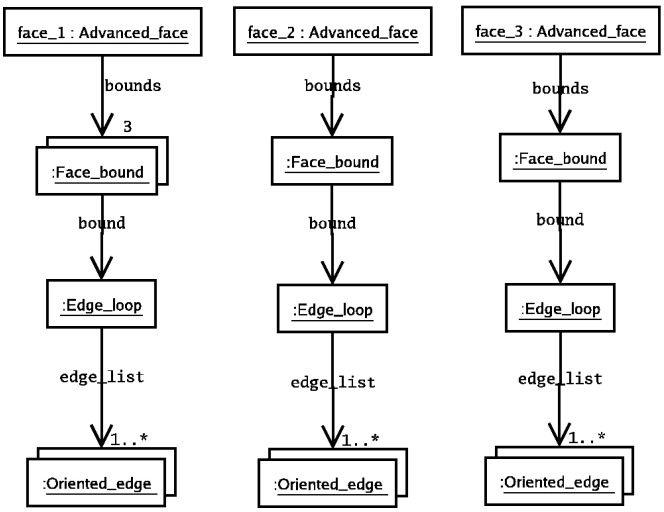


FIG. C.5 – Diagramme objets des instances Advanced_face.

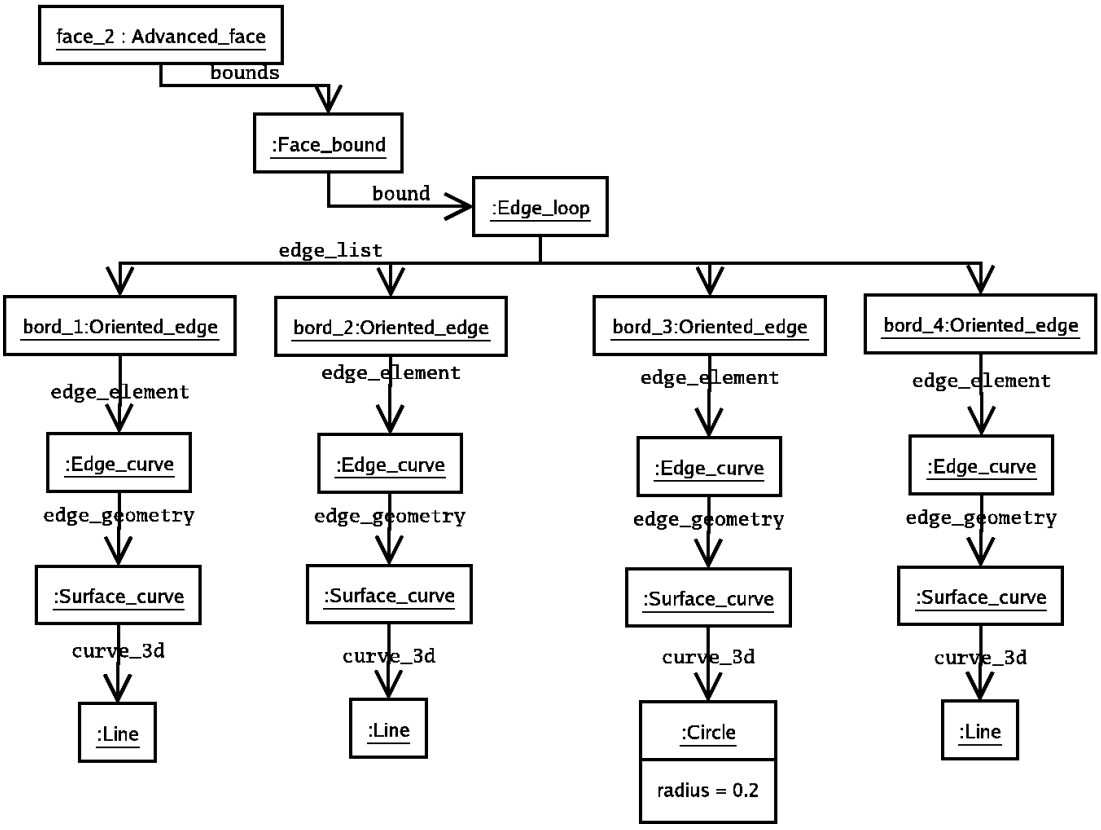


FIG. C.6 – Diagramme objets de la face 2, représentant les différents types de bords.

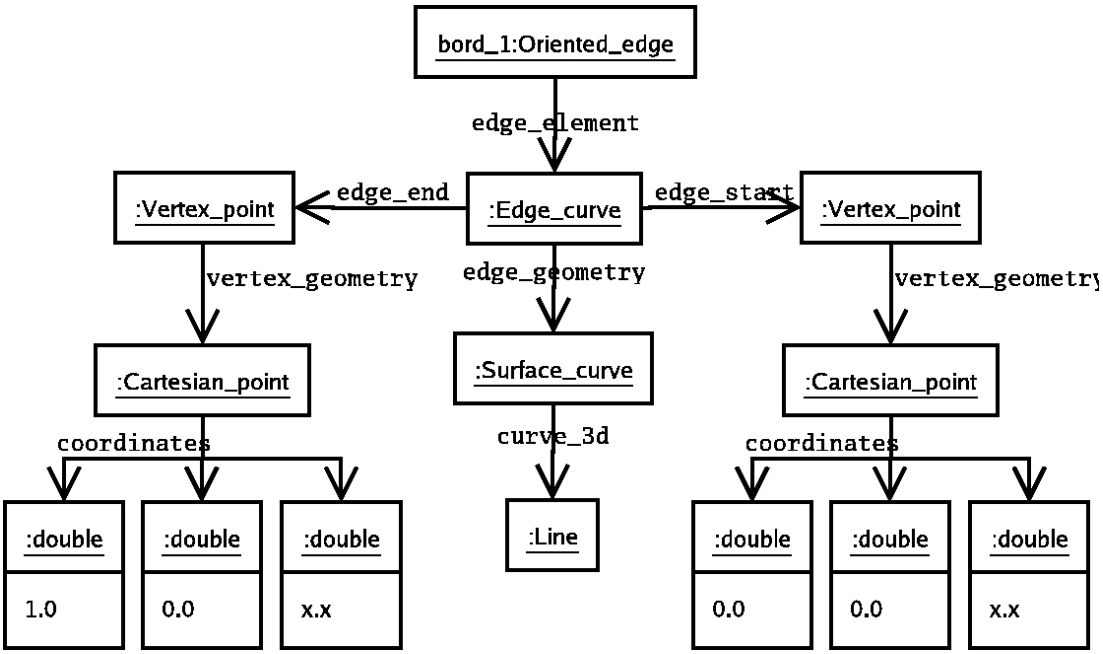


FIG. C.7 – Diagramme objets du bord 1(Segment) de la face 2 dans la représentation STEP AP214.

les coordonnées du centre du cercle (de l'arc de cercle) ; tandis que l'attribut axis nous renseigne sur le vecteur normal au cercle (information indispensable pour connaître le sens de parcours du cercle lors de sa création : Il existe deux arcs de cercle partant d'un point A et allant vers un point B).

Par ailleurs les unités des quantités sont définies dans le modèle (attribut units des instances de la classe Manifold_surface_shape_representation).

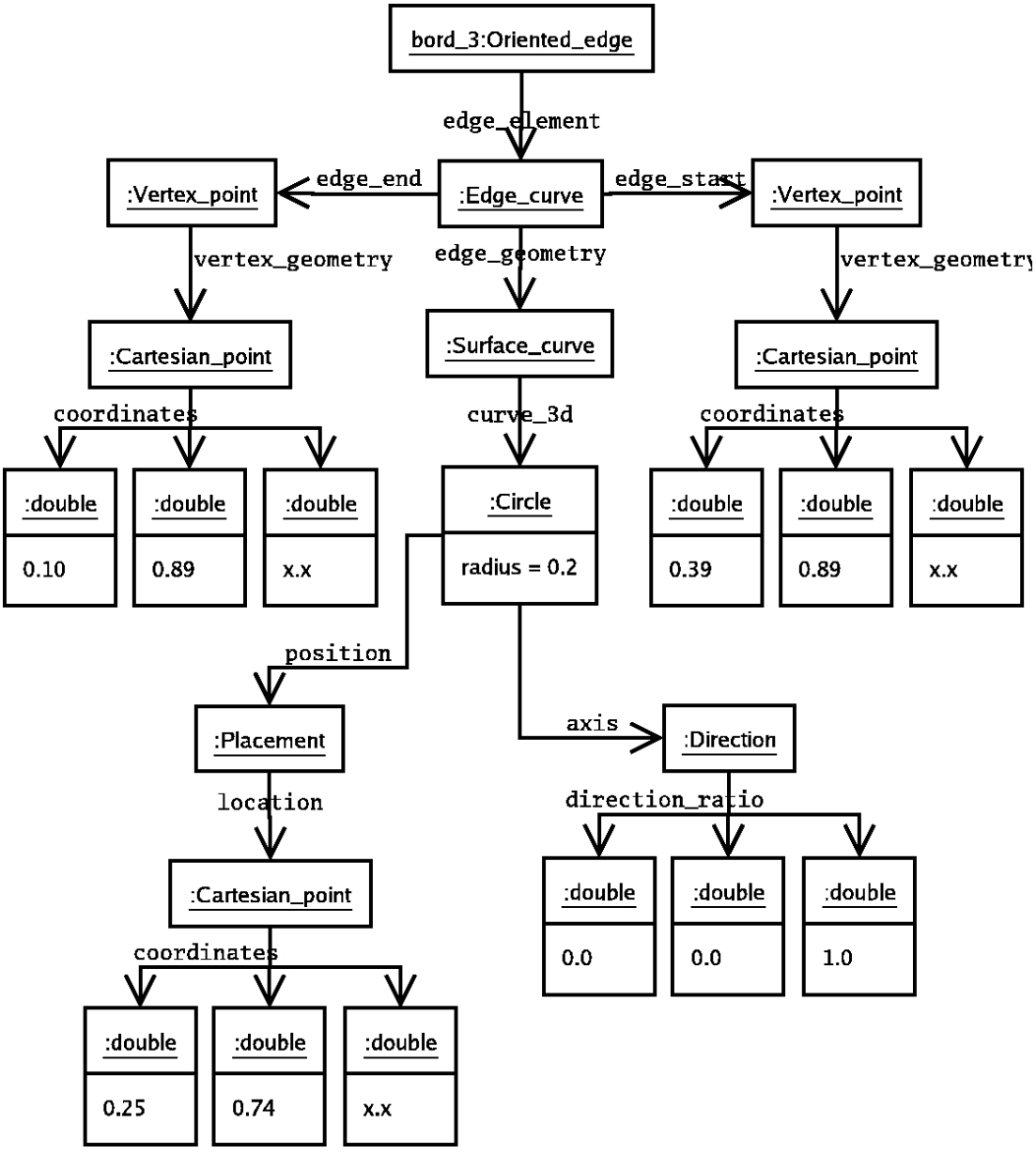


FIG. C.8 – Diagramme objets du bord 3 (Arc) de la face 2 dans la représentation STEP AP214.

Annexe D

Utilisation de XDATA

L'intérêt de SALOME pour les travaux de thèse est de disposer de composants logiciels (CAO, mailleurs, ...) déjà existants, de formats d'échange et d'un environnement graphique (IHM), distribué (CORBA) et dédié au calcul scientifique. Pour bénéficier de cet environnement, une application doit être développée sous la forme d'un composant de SALOME.

Les outils XDATA permettent de développer un ensemble de classes informatiques et de les intégrer automatiquement dans un composant SALOME. On s'affranchit alors des difficultés d'ordre technique pour l'intégration dans SALOME (interfaces IDL pour CORBA, serveur de noms, ...). Il est donc nécessaire d'étudier en détail ces outils pour réaliser l'intégration des développements de la plate-forme de modélisation dans SALOME.

La définition complète des objets XDATA est décrite dans [Ada04]. Nous verrons ici quelques exemples de création d'objets XDATA et leur intégration dans la plate-forme SALOME. Nous présenterons dans un premier temps comment sont traités les types primitifs (XType) avec XDATA puis nous nous consacrerons à la définition des classes (XObject). Enfin nous montrerons comment construire un composant SALOME avec XDATA.

D.1 Le concept XDATA

La plate-forme SALOME propose des services et des modèles de données traitant des principaux objets manipulés lors des simulations numériques : géométries, maillages, champs de valeur. La mise en donnée d'une application de simulation physique comprend d'autres données plus spécifiques et évolutives, comme les matériaux (caractéristiques physico-chimiques), les conditions aux limites d'un calcul, les données propres à une discipline qui simule le comportement d'un dispositif. Le module XDATA [Ada04] a été spécifiquement développé pour définir des modèles de données évolutifs, facilement intégrables dans SALOME. Les classes XDATA permettent ainsi de simplifier l'intégration d'un modèle de données dans la plate-forme SALOME. XDATA se présente sous la forme d'un module PYTHON et d'un ensemble d'utilitaires dédiés à la réalisation de composants XDATA pour SALOME. Dans ce cas, ces composants sont développés en

PYTHON, et leurs objets sont des instances de classes XDATA (ces objets dérivent de classes génériques XDATA).

Par ailleurs, le langage PYTHON fonctionne par typage dynamique des variables. De ce fait, ce langage admet qu'une même variable référence deux objets de types différents au cours d'une exécution. Les objets XDATA interdisent ce type d'utilisation.
Par exemple en PYTHON le code suivant est accepté :

```
>>> class A:
...     pass
>>> a = A()
>>> a.x = 3
>>> print a.x
3
>>> a.x = 'trois'
>>> print a.x
trois
```

Si la classe `A` était développée avec XDATA, alors l'affectation `a.x = 'trois'` ne serait pas permise. Ainsi la cohérence des types de données est assurée par ce module. La suite de ce paragraphe détaille l'utilisation pratique du module XDATA, dans le cadre de la création d'un module pour SALOME.

D.2 Les XType

Les XType[1] peuvent être considérés comme des testeurs de types. Le module XDATA définit les tests des types primitifs du langage PYTHON (int, float, none, string, list, tuple, dict, ...) et il est aussi possible d'en créer des nouveaux par dérivation. Pour tester une donnée, il suffit d'appeller une instance d'un XType avec cette donnée. Si la donnée correspond à l'instance de XType, elle est retournée, sinon une exception XTypeValueError est lancée. Par exemple, la classe XInt est un XType (XInt dérive de XType) et le code suivant crée des instances de XInt :

```
>>> from xdata import *
>>> x0 = XInt()
>>> x1 = XInt(min = 3 , max = 12)
>>> x2 = XInt(into = [2,3,4,5])
>>> x3 = XInt(not_into = [2,3,4,5])
```

Le test de `x0` est correct si la valeur est un entier. Le test de `x1` est correct si la valeur est un entier compris entre trois et douze. Le test de `x2` est correct si la valeur est un entier inclus dans la liste spécifiée. Le test `x3` est correct si la valeur est un entier exclus de la liste spécifiée. L'appel `x0(3)` renvoie `3` alors que `x1(2)` lance une exception `XValueError`. C'est ce mécanisme qui est à l'origine de la gestion des types des attributs des objets XDATA[2].

[1] XType est considérée comme étant une classe abstraite

[2] Contrairement au langage PYTHON utilisé sans XDATA qui permet de modifier le type d'une variable en cours d'exécution dans un programme

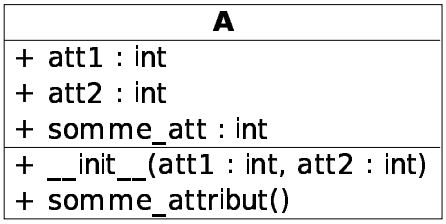


FIG. D.1 – Diagramme UML d'une classe

D.3 Les XObject

Définir une classe XDATA c'est définir une classe qui dérive de XObject. En Python la définition d'une classe A, représentée par le diagramme UML [Gir04] de la figure D.1, se rédige de la manière suivante :

```
class A:
    def __init__( init_att1=0, init_att2=0):   # constructeur
        self.att1=init_att1                     # attribut att1
        self.att2=init_att2                     # attribut att2

    def somme_attribut():                       # méthode
        self.somme_att = self.att1 +
                         self.att2          # attribut somme_att
```

Où `__init__` est le constructeur dans lequel sont initialisés les attributs `att1` et `att2`, `somme_attribut` est une méthode qui retourne la somme des deux attributs. Cette méthode crée aussi un attribut `somme_att`. Cette classe permet par exemple l'exécution suivante :

```
>>> a = A(2,5)
>>> print a.att1,a.att2
2 5
>>> a.somme_attribut()
>>> print a.somme_att
7
```

Lors de la création d'un objet XDATA, les attributs doivent être définis par des objets `XAttribute` inclus dans les listes `__init__xattributes__` et `__object__xattributes__`. Le code suivant définit les attributs `att1`, `att2` et `somme_att` en tant qu'entier :

```
xatt1       = XAttribute("att1", xtype = XInt(),
                                   default_value= 0)
xatt2       = XAttribute("att2", xtype = XInt(),
                                   default_value= 0)
xsomme_att = XAttribute("somme_att", xtype = XInt(),
                                   default_value= 0)
```

La définition de la classe A (figure D.1) en tant que classe XDATA est alors :

```
class A(XObject):  #Dérive de XObject
        __init__xattributes__    = [xatt1,xatt2]
        __object__xattributes__  = [xsomme_att]
```

```
        def somme_attribut():
                self.somme_att = self.att1 + self.att2
```

Le constructeur `__init__` est créé automatiquement à partir des attributs de la liste `__init__xattributes__`. La liste `__object__xattributes__` contient des attributs non nécessaires à la construction de l'objet. Des accesseurs pour chaque attribut sont créés automatiquement. Ainsi, lors de l'exécution, l'appel à chacun des attributs passe systématiquement par les méthodes nommées `getAtt1()`, `getAtt2()`, `getSommeAtt()`, et `setAtt1(value)`, `setAtt2(value)`, `setSommeAtt(value)`. Ces fonctions peuvent être surchargées et la classe `A` peut être simplifiée de la sorte :

```
class A(XObject):   #Dérive de XObject
        __init__xattributes__    = [xatt1,xatt2]
        __object__xattributes__ = [xsomme_att]
        def getSommeAtt():
                return   self.att1 + self.att2
```

La méthode `somme_attribut` n'est alors plus nécessaire. En effet, lors du simple appel de l'attribut `somme_att`,le mécanisme XDATA utilise automatiquement l'accesseur `getSommeAtt()`. Le code se résume à ceci :

```
>>> a = A(2,5)
>>> print a.att1,a.att2
2 5
>>> print a.somme_att
7
```

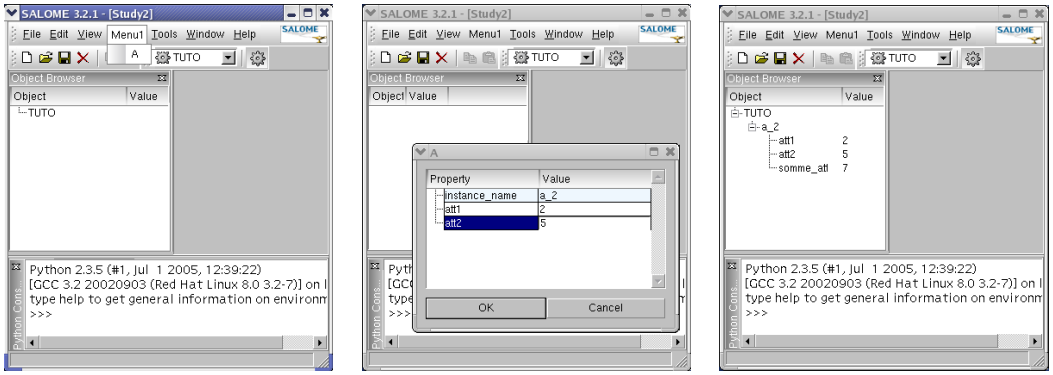
D.4 Contruction d'un composant SALOME avec XDATA

Un composant SALOME développé avec XDATA est principalement un ensemble de classes qui dérivent de la classe XObject, ainsi que des descriptions de menus et d'items spécifiques à l'utilisation d'une interface graphique.

La *grammaire* XDATA est interprétée par des outils XDATA, qui construisent les interfaces nécessaires à CORBA. Ces interfaces sont sous forme de fichier *idl*. Elles définissent les services proposés par le composant et permettent de communiquer avec les divers autres composants de SALOME.

En pratique, lorsque qu'une architecture logicielle est encapsulée en un composant XDATA, l'interface graphique de SALOME permet d'agir sur les instances des différentes classes de cette architecture. L'utilisateur a alors la possiblité de créer des objets, de modifier des attributs et d'appeler des méthodes. L'interface graphique est modifiée au niveau de la barre de menu qui devient spécifique au composant.

Intégrer un composant dans SALOME suppose donc de définir des menus d'accès aux classes. Par exemple, pour intégrer un composant nommé *TUTO* qui contient la classe *A* dans un menu *menu1*, il est nécessaire de créer un fichier PYTHON nommé *TUTO_xdata.py* contenant les lignes de code suivantes :



(a) Le menu menu1 contient l'item A

(b) Instanciation de la classe A

(c) Les attributs de l'objet sont visibles, l'attribut somme_att a été calculé

FIG. D.2 – Intégration d'un composant TUTO contenant la classe A.

```
__xdata__items__ = [
          "menu1",
          ]
```

Ainsi que le fichier *menu1.py* qui décrit les items du menu et les classes :

```
__xdata_items__ = [
          'A',
          ]

from xdata import *

xatt1       = XAttribute("att1",xtype = XInt(), default_value=0)
xatt2       = XAttribute("att2",xtype = XInt(), default_value=0)
xsomme_att = XAttribute("somme_att", xtype = XInt())

class A(XObject):
    __init__xattributes__    =[xatt1,xatt2]
    __object__xattributes__  =[xsomme_att]

    def getSommeAtt(self):
        return self.att1+self.att2
```

Annexe E

Description des principales matrices de transformations géométriques

En pratique, une transformation géométrique peut-être décrite via une matrice τ_{ra}. Le point P' image de P par la transformation décrite par τ_{ra} est défini comme suit :

$$P' = [\tau_{ra}] \cdot P$$

Généralement[1], ces opérations sont réalisées en coordonnées homogènes, c'est-à-dire dans un espace de dimension $n+1$. Ainsi, un point de coordonnées (x, y, z) correspond à $(x, y, z, 1)$. Dans le cas de géométries placées dans un espace de dimension 3, τ_{ra} est une matrice 4×4. Les matrices de transformation usuelles sont définies ci-dessous :

– La translation de vecteur (T_x, T_y, T_z) :

$$\tau_{ra} = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

– La matrice de dilatation de centre (C_x, C_y, C_z) et de coefficients $(\alpha_x, \alpha_y, \alpha_z)$:

$$\tau_{ra} = \begin{bmatrix} \alpha_x & 0 & 0 & C_x\,(1-\alpha_x) \\ 0 & \alpha_y & 0 & C_y\,(1-\alpha_y) \\ 0 & 0 & \alpha_z & C_z\,(1-\alpha_z) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

– La matrice de rotation d'angle β autour de l'axe des x et de centre l'origine :

$$\tau_{ra} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta & 0 \\ 0 & \sin\beta & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

– La composition de transformations géométriques n'est autre que le produit des matrices de transformation. Ainsi, une rotation d'angle β autour de l'axe des x et de

[1] Afin de pouvoir travailler sur des points placés à l'infini et de ne pas effectuer des traitements particuliers pour le parallélisme.

centre (P_x, P_y, P_z) peut être réalisée en composant trois transformations : Une translation de vecteur $(-P_x, -P_y, -P_z)$, suivie d'une rotation d'angle β autour de l'axe des x et de centre l'origine, et enfin une translation de vecteur (P_x, P_y, P_z). Cette composition est représentée par l'opération suivante :

$$\tau_{ra} = \begin{bmatrix} 1 & 0 & 0 & P_x \\ 0 & 1 & 0 & P_y \\ 0 & 0 & 1 & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta & 0 \\ 0 & \sin\beta & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & -P_x \\ 0 & 1 & 0 & -P_y \\ 0 & 0 & 1 & -P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

– La matrice de rotation d'angle α, d'axe (A_x, A_y, A_z) et de centre $(0, 0, 0)$ (tirée de [FG99]) :

$$\tau_{ra} = \begin{bmatrix} a^2 - Sd + CS_2g & ab - Se + CS_2h & ac - Sf + CS_2i & 0 \\ SC_2a + Cd + SS_2g & SC_2b + Ce + SS_2h & SC_2c + Cf + SS_2i & 0 \\ -S_2a + C_2g & -S_2b + C_2h & -S_2c + C_2i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

En considérant,

– si $A_x \neq 0$, $\phi = \arctan\left(\frac{A_y}{A_x}\right)$, $\theta = \arctan\left(\frac{A_z}{\sqrt{A_x^2 + A_y^2}}\right)$, $C = \cos\phi$ et $S = \sin\phi$, $C_2 = \cos\theta$ et $S_2 = \sin\theta$,

– sinon, si $A_y \neq 0$, $\theta = \arctan\left(\frac{A_z}{\sqrt{A_x^2 + A_y^2}}\right)$, $C = 0$, $S = 1$, et $C_2 = \cos\theta$, $S_2 = \sin\theta$,

– sinon $C = 0$, $S = 1$ et $C_2 = 0$, $S_2 = 1$.

avec $a = C_2C$, $b = C_2S$, $c = -S_2$ et $d = -\cos(\alpha)\,S - \sin(\alpha)\,S_2C$, $e = \cos(\alpha)\,C - \sin(\alpha)\,SS_2$ et $f = -\sin(\alpha)\,C_2$ et $g = -sin(\alpha)\,S + \cos(\alpha)\,S_2C$, $h = \sin(\alpha)\,C + \cos(\alpha)\,SS_2$ et $i = \cos(\alpha)\,C_2$.

Annexe F

Description détaillée d'une classe Circle.

Dans le paragraphe 7.1.3, les principales méthodes d'une classe `Circle` sont présentées. Toutefois, de manière à favoriser la compréhension, l'implémentation de cette classe, illustrée par la figure F, est simplifiée. En effet, même si elle permet l'application des principales transformations géométriques, les dilatations ne sont pas permises. Ceci est dû au fait que l'attribut `radius` de la classe `Circle` du paragraphe 7.1.3 n'est pas dépendant d'objets de `Point` (A contrario les attributs `center` et `axis` sont soit des objets `Point` soit en sont dépendants), retrouvé dans la liste `spacePoints`.

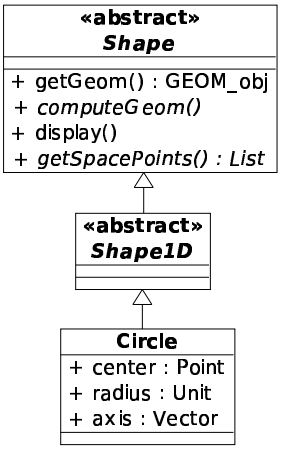


FIG. F.1 – Diagramme de classe de Circle

Nous détaillons dans cette annexe une implémentation possible de la classe `Circle`.

Dans l'implémentation suivante, l'objectif est de faire dépendre l'attribut `radius` d'objets `Point`. La solution proposée consiste à ajouter un attribut (considéré comme privé), nommé `vectorForRadius`. Cet attribut correspond à un objet `vector` qui représente un des rayons du cercle, orthogonal à l'axe du cercle. `radius` est dans ce cas le résultat du calcul de la norme de ce vecteur. Les deux points caractérisant ce vecteur sont ajoutés à la liste `spacePoints`. Ainsi une dilatation du cercle implique une dilatation de ce vecteur et donc une modification de la valeur du rayon.

Nous rappelons que tout appel en lecture et en écriture, aux attributs d'un `XObject` font systématiquement appel de manière transparente à des accesseurs surchargeables *set* et *get*. Ainsi, si `c` est un objet `Circle`, alors *c.radius = 3* est équivalent à *c.setRadius(3)* ; de même *a = c.radius* est équivalent à *a = c.getRadius()*.

Plusieurs cas d'utilisation d'un objet `Circle` sont distingables :

- lors de la modification de l'attribut `radius` (`setRadius`), le vecteur représentant un rayon doit être recalculé. Etant donné que ce vecteur dépend de même de l'axe du cercle et que nous avons choisi de le positionné au centre du cercle, les modifications de ces deux autres attributs impliquent aussi un nouveau calcul de ce vecteur. Le calcul des composantes de ce vecteur est réalisé par la méthode définie à la ligne 17. Cette méthode est appelée dans `update()` ;

- lors d'une transformation géométrique, les deux points `vectorForRadius` sont modifiés. Ainsi, l'accesseur `getRadius` perttra de retourner la norme de ce vecteur ;

- lors de la contruction de l'objet `Circle`, les attributs de construction, contenus dans `__init__xattributes__` , sont initialisés aux valeurs passées en arguments du constructeur. Les accesseurs de ces attributs sont appelés, dont les premiers appels à `setRadius`. Comme il n'y a pas suffisament d'information pour calculer le vecteur rayon, ce n'est que le premier appel à `getRadius` qui permettra de construire ce vecteur (ligne 76).

```
1  from xdata import *
2  from shape1D import Shape1D
3  from point import Point
4  from vector import Vector
5
6  class Circle(Shape1D):
7      __init__xattributes__ = [
8          XAttribute("radius", xtype=XFloat(open_min=0.0)),
9          XAttribute("center",xtype=XInstance(Point)),
10         XAttribute("axis",xtype=XInstance(Vector)),
11         ] + Shape1D.__init__xattributes__
12
13     __object__xattributes__ = [
14         XAttribute("vectorForRadius", xtype = XInstance(Vector),mode='r'),
15         ] + Shape1D.__object__xattributes__
16
17     def computeRadiusVectorComponents(self,R):
18         axis = self.axis
19         x,y,z = axis.x,axis.y,axis.z
20         print "x,y,z",x,y,z
21         from math import sqrt
22         if x!= 0. and y != 0. :
23                 c = 0
24                 b = R / sqrt( 1 + (y/x)**2 )
25                 a = -b * y/x
```

```
26          elif x!= 0 :
27                  b = 0
28                  c = R / sqrt( 1 + (z/x)**2 )
29                  a = -c * z/x
30          elif y!= 0 :
31                  a = 0
32                  c = R / sqrt( 1 + (z/y)**2 )
33                  b = -c * z/y
34          else :
35                  a = 0
36                  b = R / sqrt( 1 + (y/z)**2 )
37                  c = -b * y/z
38          return a,b,c
39
40      def setRadius(self,value):
41          try:
42                  radius , vfr,axis , center =
43                                  self.radius,
44                                  self.vectorForRadius,
45                                  self.axis,
46                                  self.center
47          except AttributeError:
48                  return
49          a,b,c = self.computeRadiusVectorComponents(value)
50          vfr.point1 = center
51          vfr.point2.x = center.x + a
52          vfr.point2.y = center.y + b
53          vfr.point2.z = center.z + c
54          if self.displayed :
55                  self.computeGeom()
56          pass
57
58      def setSomething(self, *args, **kwargs):
59          self.update()
60          return
61
62      setCenter    = setSomething
63      setAxis      = setSomething
64      setDisplayed = setSomething
65
66      def getRadius(self):
67          try :
68                  vfr = self.vectorForRadius
69                  from math import sqrt
70                  length = sqrt(vfr.x**2 + vfr.y**2 + vfr.z**2)
71                  return length
72          except AttributeError :
73                  center = self.center
74                  a,b,c = self.computeRadiusVectorComponents(self.radius)
75                  p = Point(center.x + a, center.y + b, center.z + c)
```

```
76                  self.vectorForRadius = Vector(center,p)
77                  return self.radius
78          return
79
80      def getSpacePoints(self):
81          return self.center.getSpacePoints() +
82                  self.axis.getSpacePoints() +
83                  self.vectorForRadius.getSpacePoints()
84
85      def update(self):
86          try:
87              radius, center, axis, vfr, dislayed =
88                                      self.radius,
89                                      self.center,
90                                      self.axis,
91                                      self.vectorForRadius,
92                                      self.displayed
93          except AttributeError:
94              return
95          a,b,c = self.computeRadiusVectorComponents(radius)
96          vfr.point1 = center
97          vfr.point2.x = center.x + a
98          vfr.point2.y = center.y + b
99          vfr.point2.z = center.z + c
100         if self.displayed :
101             self.computeGeom()
102         return
103
104     def computeGeom(self):
105         import batchmode_geompy
106         b_geompy = batchmode_geompy
107         radius = self.radius
108         center = self.center
109         axis = self.axis
110         geom_obj=b_geompy.MakeCircle(center.getGeom(),
111                                      axis.getGeom(),
112                                      radius)
113         self.geom_obj = geom_obj
114         return
115
116     ...
```

Annexe G

Création d'un cube à l'aide du composant GEOM

Cette annexe présente la construction en PYTHON d'un cube unité à l'aide du composant GEOM.

La construction ci-dessous consiste dans un premier temps à créer les huit sommets, puis les douze arêtes. Cela consiste en des appels aux méthodes `MakeVertex(...)` et `MakeLineTwoPnt(...)` du composant GEOM (module PYTHON `geompy`). Les commandes `geompy.addToStudy(...)` permettent d'associer un nom à l'objet géométrique et à ajouter cet objet dans l'interface d'étude de SALOME. Ces commandes retournent un identifiant pour chaque objet ajouté à l'étude.

```
1  import geompy
2  l_vertex = []
3  l_vertex.append(geompy.MakeVertex(0,0,0))
4  l_vertex.append(geompy.MakeVertex(1,0,0))
5  l_vertex.append(geompy.MakeVertex(1,1,0))
6  l_vertex.append(geompy.MakeVertex(0,1,0))
7
8  l_vertex.append(geompy.MakeVertex(0,0,1))
9  l_vertex.append(geompy.MakeVertex(1,0,1))
10 l_vertex.append(geompy.MakeVertex(1,1,1))
11 l_vertex.append(geompy.MakeVertex(0,1,1))
12
13 l_id_vertex = [geompy.addToStudy(l_vertex[i],"v%d"%i )
14                          for i in range(len(l_vertex))]
15
16 l_line = []
17 l_line.append(geompy.MakeLineTwoPnt(l_vertex[0],l_vertex[1]))
18 l_line.append(geompy.MakeLineTwoPnt(l_vertex[1],l_vertex[2]))
19 l_line.append(geompy.MakeLineTwoPnt(l_vertex[2],l_vertex[3]))
20 l_line.append(geompy.MakeLineTwoPnt(l_vertex[3],l_vertex[0]))
21
22 l_line.append(geompy.MakeLineTwoPnt(l_vertex[0],l_vertex[4]))
23 l_line.append(geompy.MakeLineTwoPnt(l_vertex[1],l_vertex[5]))
24 l_line.append(geompy.MakeLineTwoPnt(l_vertex[2],l_vertex[6]))
```

```
25 l_line.append(geompy.MakeLineTwoPnt(l_vertex[3],l_vertex[7]))
26
27 l_line.append(geompy.MakeLineTwoPnt(l_vertex[4],l_vertex[5]))
28 l_line.append(geompy.MakeLineTwoPnt(l_vertex[5],l_vertex[6]))
29 l_line.append(geompy.MakeLineTwoPnt(l_vertex[6],l_vertex[7]))
30 l_line.append(geompy.MakeLineTwoPnt(l_vertex[7],l_vertex[4]))
31
32 l_id_line = [geompy.addToStudy(l_line[i],"l%d"%i)
33                             for i in range(len(l_line))]
```

Dans un second temps, les bords (ou wire) des six faces sont construits à partir des arêtes. Pour cela les arêtes sont groupées quatre par quatre dans des listes. Ces listes sont passées en argument de la méthode `MakeWire(...)`.

```
34 l_wire = []
35 l_wire.append(geompy.MakeWire(l_line[0:4]))
36 l_wire.append(geompy.MakeWire(l_line[-4:]))
37 l_wire.append(geompy.MakeWire([
38                             l_line[0],
39                             l_line[5],
40                             l_line[8],
41                             l_line[4]]))
42 l_wire.append(geompy.MakeWire([
43                             l_line[1],
44                             l_line[6],
45                             l_line[9],
46                             l_line[5]]))
47 l_wire.append(geompy.MakeWire([
48                             l_line[2],
49                             l_line[7],
50                             l_line[10],
51                             l_line[6]]))
52 l_wire.append(geompy.MakeWire([
53                             l_line[3],
54                             l_line[4],
55                             l_line[11],
56                             l_line[7]]))
57
58 l_id_wire = [geompy.addToStudy(l_wire[i],"w%d"%i)
59                             for i in range(len(l_wire))]
```

Les faces peuvent maintenant être définies à partir de bords fermés. La surface (shell) du cube est ensuite construite à partir des six faces. Enfin le cube (box) est construit comme étant un solide (un volume) à partir de sa surface (fermée).

```
60 isPlanarWanted = True
61 l_face = [geompy.MakeFaces([wire], isPlanarWanted)
62                             for wire in l_wire]
63
64 l_id_face = [geompy.addToStudy(l_face[i] , "f%d"%i)
65                             for i in range(len(l_face))]
66
```

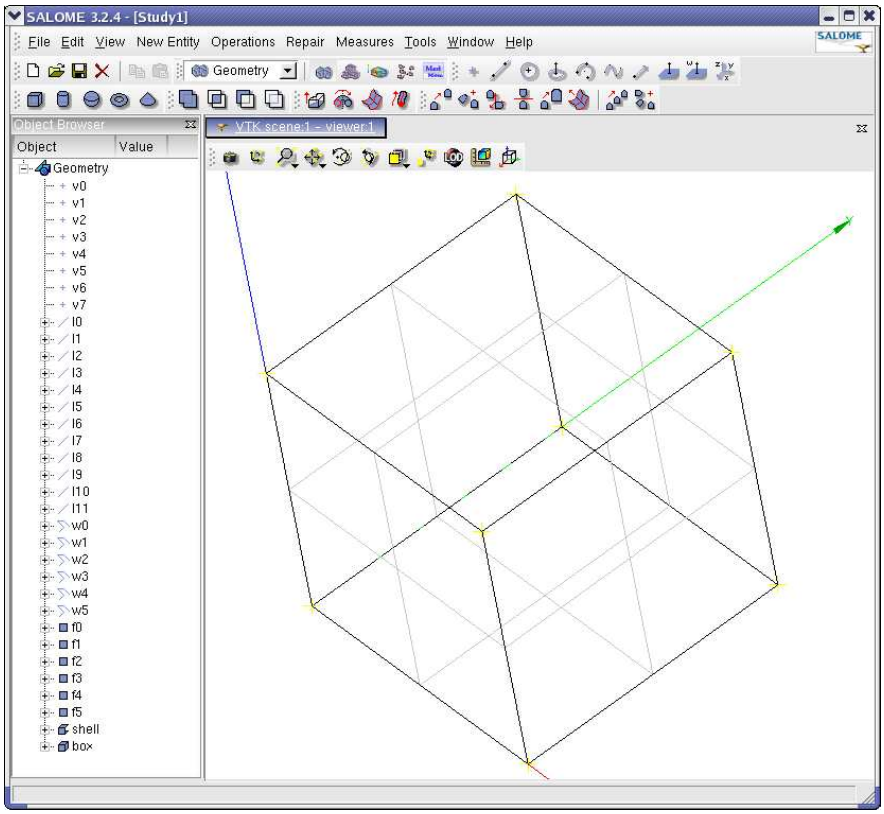


FIG. G.1 – Cube unitaire et ses entités, construit par le composant GEOM.

```
67 shell = geompy.MakeShell(l_face)
68 id_shell = geompy.addToStudy(shell,'shell')
69
70 box = geompy.MakeSolid([shell])
71 id_box =  geompy.addToStudy(box,'box')
```

Pour terminer, nous prenons le soin d'afficher ces objets dans SALOME :

```
73 import salome
74 gg = salome.ImportComponentGUI("GEOM")
75 for id_obj in l_id_vertex +
76                l_id_line +
77                l_id_wire +
78                l_id_face +
79                [id_shell] +
80                [id_box] :
81     gg.createAndDisplayGO(id_obj)
```

Le résultat de l'exécution de ce script est présenté par la figure G.1.

Annexe H

Description des menus du composant TECHENTITY.

Les entités géométriques sont accessibles par le menu *Geometry* (figure H.1). Elles sont classées selon leur dimension. Une catégorie d'entités *Spécial* a été ajoutée.

FIG. H.1 – Menu de géométrie

Le menu des entités géométriques de dimension 0 ne contient qu'un point.

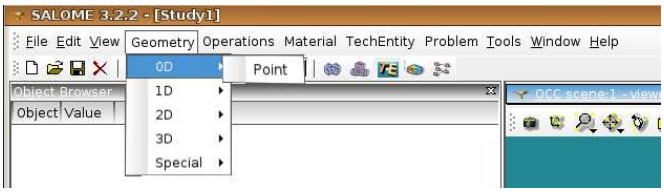


FIG. H.2 – Menu géométrie 0D

Le menu des entités géométriques de dimension 1 (figure H.3) permet de construire des segments, des arcs de cercles et des cercles, des vecteurs, des courbes (*Curve*) de type B-Spline à partir d'une liste de points, et des courbes composées (*Wire*) d'une liste entités de dimension 1. Ces courbes composées peuvent par exemple être le périmètre d'une surface carrée.

Le menu des entités géométriques de dimension 2 (figure H.4) permet de construire des surfaces telles que des disques (*Disc*) ou des faces planes (*Face*) à partir d'un ensemble fermé de courbes (*Wire)*, et des *Shells* qui représentent une composition de faces qui ne sont pas nécessairement coplanaires.

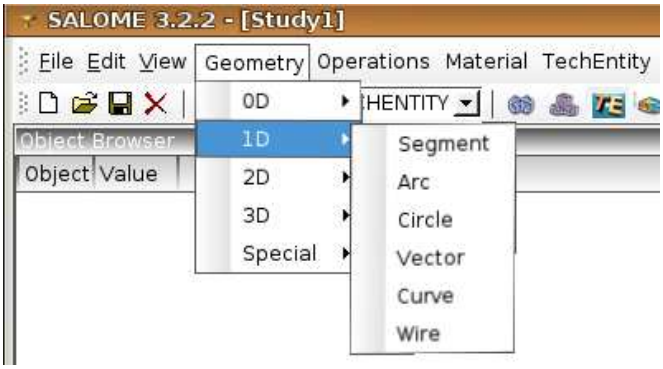


FIG. H.3 – Menu géometrie 1D

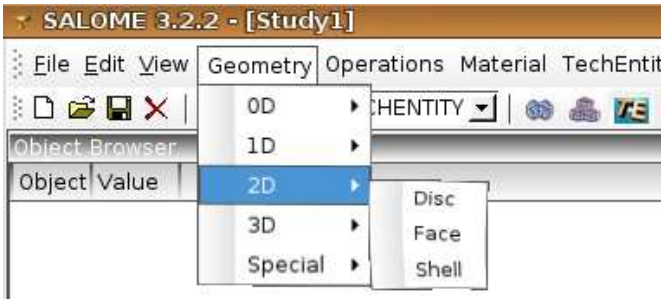


FIG. H.4 – Menu géometrie 2D

Le menu des entités géométriques de dimension 3 (figure H.5) permet de construire des volumes tels que des parallélépipèdes rectangles (*Box*), des cylindres et des boules (*Sphere*), des tubes (cylindres creux), des parallélépipèdes rectangles aux coins *arrondis*, et des solides quelconques à partir d'un ensemble de faces (*Shell*) fermé.

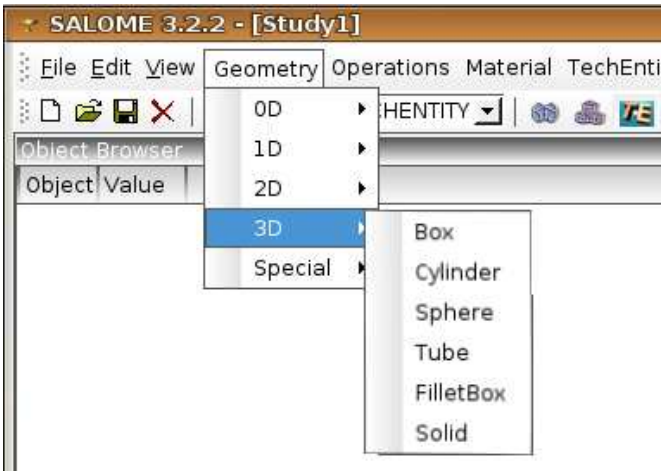


FIG. H.5 – Menu géometrie 3D

Le menu des entités géométriques *Special* (figure H.6) permet de construire trois types particuliers objets :

– *PyGeomFile* permet de construire un objet quelconque à partir d'un script en PY-

THON (plus de détails sur son utilisation sont disponibles en annexe J) ;
- *TransformGeometry* permet de construire, à partir d'une entité géométrique quelconque et d'une transformation géométrique, l'entité géométrique image par cette transformation ;
- *GeomObjectGeometry* permet de *voir* un objet du composant GEOM comme étant une entité géométrique du composant TECHENTITY.

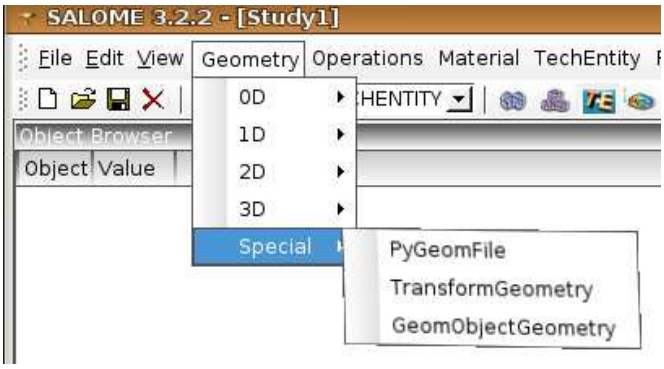


FIG. H.6 – Menu géometrie Special

Actuellement, le menu *Operations* (figure H.7) ne contient que des transformations géométriques.

- *TransformMatrix* permet de définir une matrice de transformation 4×4 ;
- *TranslateMatrix* permet de définir une matrice de translation ;
- *RotateMatrix* permet de définir une matrice de rotation ;
- *ScaleMatrix* permet de définir une matrice de dilatation ;
- *Transformation* permet de définir une composition de transformations.

Ces matrices sont détaillées dans l'annexe E.

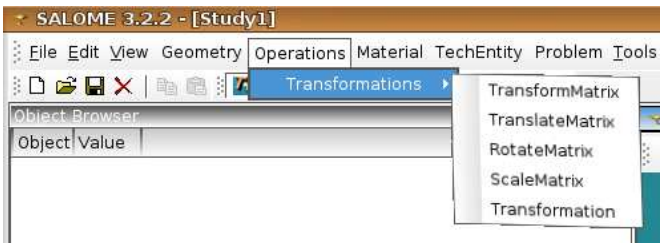


FIG. H.7 – Menu des transformations

Le menu des materiaux *Material* (figure H.8) permet de construire des objets définissant des Matériaux caractérisés selon qu'ils sont liquides (*Fluide*), gazeux (*Gaz*), ou bien que ce sont des matériaux solides de structures ou combustibles.

Le menu *TechnologyEntity* (figure H.9) permet de construire des entités technologiques.

Le menu des problèmes (figure H.10) permet de construire les interfaces des différents types de problèmes (figures H.11 et H.12).

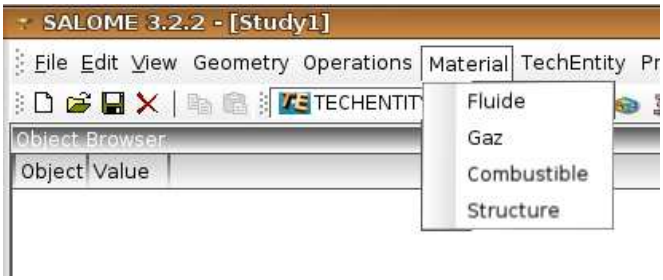


FIG. H.8 – Menu des materiaux

FIG. H.9 – Menu des Entités Technologiques

FIG. H.10 – Menu des problèmes

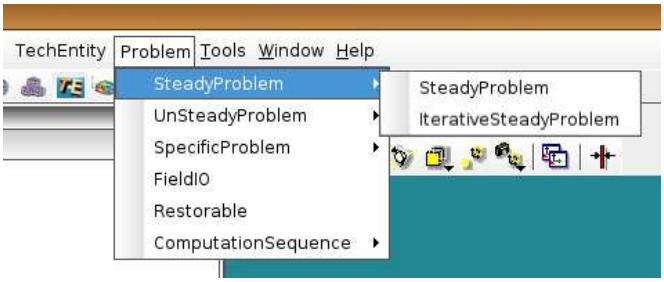


FIG. H.11 – Menu des problèmes stationnaires

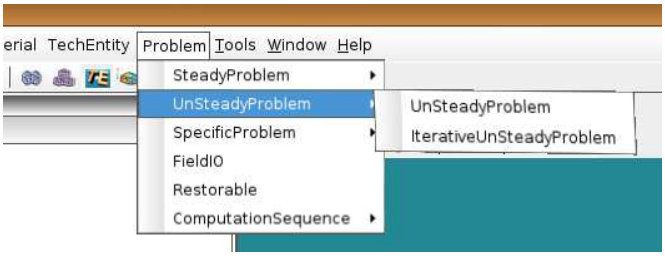


FIG. H.12 – Menu des problèmes instationnaires

Le menu des problèmes spécifiques (figure H.13) présente des schémas de calcul utili-

sables par des objets problèmes (préalablement encapsulés).

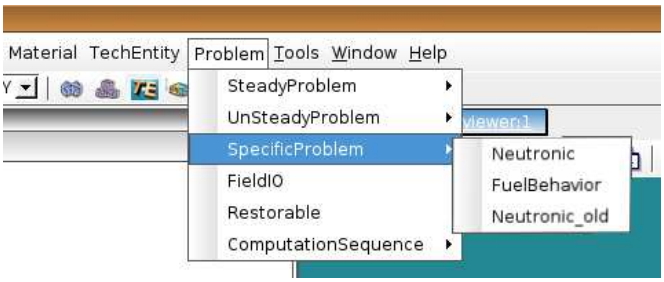


FIG. H.13 – Menu des problèmes spécifiques

Annexe I

Création d'un cube à l'aide des objets Shape du composant TECHENTITY

Cette annexe présente la construction en PYTHON d'un cube unité à l'aide des objets `Shape` du composant TECHENTITY.

Un cube unité, donc de dimension 3, peut être construit de deux manières. Soit en créant un objet `Box`, accessible par le sous-menu *3D* du menu *Geometry* (voir annexe H), ou au travers du script python suivant :

```
from box import Box
box_1 = Box(dx=1.,dy=1.,dz=1.)
```

Soit à partir d'objets géométriques de dimensions inférieures, au travers de l'interface graphique de SALOME ou bien d'un script en PYTHON équivalent :

- Dans un premier temps, sont construits les huit points représentant les sommets du cube :

```
from point import Point

point_1=Point(x=0.0,y=0.0,z=0.0)
point_2=Point(x=1.0,y=0.0,z=0.0)
point_3=Point(x=1.0,y=1.0,z=0.0)
point_4=Point(x=0.0,y=1.0,z=0.0)

point_11=Point(x=0.0,y=0.0,z=1.0)
point_21=Point(x=1.0,y=0.0,z=1.0)
point_31=Point(x=1.0,y=1.0,z=1.0)
point_41=Point(x=0.0,y=1.0,z=1.0)
```

- Les douze arêtes sont alors définies à partir de ces points :

```
from segment import Segment

segment_1=Segment(point1=point_1,point2=point_2)
segment_2=Segment(point1=point_2,point2=point_3)
```

```
segment_3=Segment(point1=point_3,point2=point_4)
segment_4=Segment(point1=point_4,point2=point_1)

segment_11=Segment(point1=point_11,point2=point_21)
segment_21=Segment(point1=point_21,point2=point_31)
segment_31=Segment(point1=point_31,point2=point_41)
segment_41=Segment(point1=point_41,point2=point_11)

segment_1c=Segment(point1=point_11,point2=point_1)
segment_2c=Segment(point1=point_21,point2=point_2)
segment_3c=Segment(point1=point_31,point2=point_3)
segment_4c=Segment(point1=point_41,point2=point_4)
```

– Pour définir les six faces, il est nécessaire de définir les bords(`Wire`) délimitant chacune d'entre elles :

```
from wire import Wire
from face import Face

wire_1=Wire(edges=[segment_1,segment_2,segment_3,segment_4])
face_1=Face(wires=[wire_1])
wire_2=Wire(edges=[segment_11,segment_21,segment_31,segment_41])
face_2=Face(wires=[wire_2])
wire_3=Wire(edges=[segment_1,segment_11,segment_1c,segment_2c])
face_3=Face(wires=[wire_3])
wire_4=Wire(edges=[segment_2,segment_21,segment_2c,segment_3c])
face_4=Face(wires=[wire_4])
wire_5=Wire(edges=[segment_3,segment_31,segment_3c,segment_4c])
face_5=Face(wires=[wire_5])
wire_6=Wire(edges=[segment_4,segment_41,segment_4c,segment_1c])
face_6=Face(wires=[wire_6])
```

– De la même manière, pour définir le volume (`Solid`), une *coquille* (`Shell`) correspondant à l'ensemble des surfaces le composant est définie :

```
from shell import Shell
from solid import Solid
shell_1 = Shell(faces=[face_1,face_2,face_3,face_4,face_5,face_6])
solid_1 = Solid(shells = shell_1)
```

Annexe J

Utilisation d'un objet PyGeomFile

Cette annexe présente un exemple d'utilisation d'un objet PyGeomFile.

Supposons qu'il existe le script suivant permettant de construire un objet GEOM.

```
if __name__ != '__main__':
    #variables
    from point import Point
    rayon = 2.5
    height:10
    p1 = Point(3,4,5)
    p2 = Point(1,-1,2)

    #construction
    from geompy import *
    v1 = MakeVertex(p1.x,p1.y,p1.z)
    v2 = MakeVertex(p2.x,p2.y,p2.z)
    v12 = MakeVector(v1,v2)
    b  = MakeBoxTwoPnt(v1,v2)
    c  = MakeCylinder(v1,v12,rayon,height)
    #objet GEOM
    geom_obj = MakeBoolean(b,c,2)
```

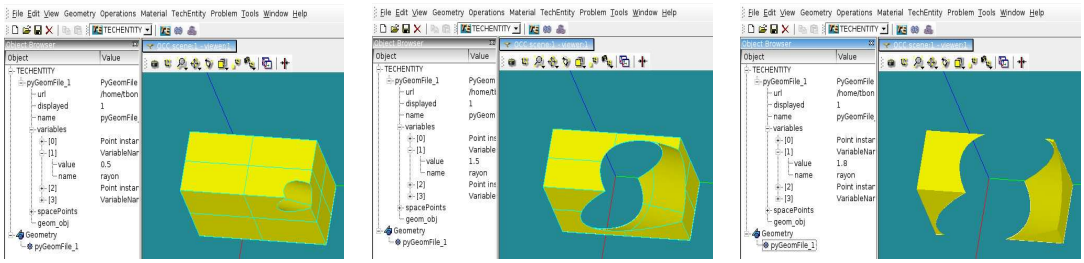
La classe `PyGeomFile` permet d'encapsuler ce script dans un objet `Shape`, moyennant quelques adaptations. L'objet `PyGeomFile` peut ensuite être utilisé en tant que géométrie locale d'une Entité Technologique.
Les adaptations de ce script consistent à repérer les variables à paramétrer et les points caractéristiques, supports des transformations géométriques. Ces variables doivent être définies dans un dictionnaire nommé `variables`, accessible uniquement lors d'une exécution du script en programme principal (lignes 1 à 6 du script ci-dessous).

```
1 if __name__ != '__main__':
2         from point import Point
3         variables = {"rayon":2.5,
4                 'height':10,
5                 'p1':Point(3,4,5),
6                 'p2':Point(1,-1,2)}
```

```
7
8
9  cmd = """
10 v1 = MakeVertex(p1.x,p1.y,p1.z)
11 v2 = MakeVertex(p2.x,p2.y,p2.z)
12 v12 = MakeVector(v1,v2)
13 b  = MakeBoxTwoPnt(v1,v2)
14 c  = MakeCylinder(v1,v12,rayon,height)
15 geom_obj = MakeBoolean(b,c,2)
16 print geom_obj
17
18 """
```

La partie *Construction* n'est pas modifiée, mais considérée comme une chaîne de caractères attribuée à la variable `cmd`[1]. Il convient d'attribuer l'objet GEOM à la variable nommée `geom_obj` en fin de script.

L'objet `PyGeomFile` ainsi créé peut être paramétré par les objets de la liste `variables` de `PyGeomFile`, que ce soit par script, ou bien avec l'aide de l'interface graphique SALOME (figure J.1).



(a) rayon = 0,5 (b) rayon = 1,5 (c) rayon = 1,8

FIG. J.1 – Variation du paramètre *rayon* de l'objet *PyGeomFile*.

[1] la partie construction pourrait tout aussi être incluse dans un autre fichier, dont la lecture serait attibuée à la variable *cmd*.

Annexe K

Construction d'un modèle d'Entités Technologiques du dispositif TANOXOS

Cette annexe présente un exemple de script de construction du dispositif TANOXOS. Le résultat de ce script est présenté par la figure K.1.

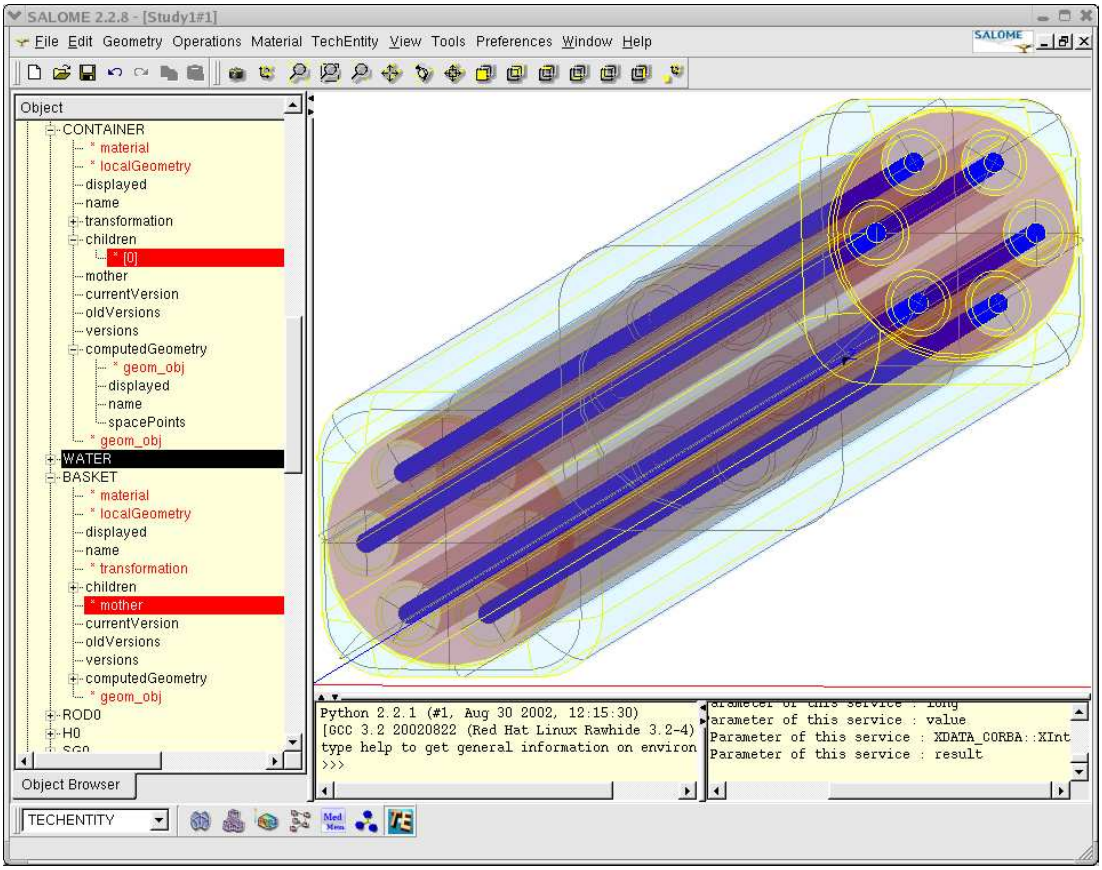


FIG. K.1 – Visualisation du dispositif TANOXOS modélisé par un arbre d'Entités Technologiques.

```
from point import Point
```

```
from vector import Vector
from filletBox import FilletBox
from cylinder import Cylinder
from transformMatrix import *

Oo       = Point(x=0.0              ,
                 y=0.0              ,
                 z=0.0    )
Om       = Point(x=7.7000000000000002     ,
                 y=7.7000000000000002     ,
                 z=60.0   )
center   = Point(x=3.8500000000000001     ,
                 y=3.8500000000000001     ,
                 z=0.0    )
OoZ1     = Point(x=0.0              ,
                 y=0.0              ,
                 z=1.0    )
vz       = Vector(point1 = Oo, point2 = OoZ1)
ptr      = Point(6.05,3.85,0)
vtr      = Vector(Oo, ptr )
translateRod = TranslateMatrix(vector = vtr)
rotateRod    = RotateMatrix(axis = vz,center =center,alpha = 60.)

from material import Material
ALU_FOURREAU = Material()
ALU_BARILLET = Material()
EAU_FOU_BAR  = Material()
EAU_BAR_CR   = Material()
SURGAINE     = Material()
GAINE        = Material()
UO2          = Material()

from technologyentity import TechnologyEntity
Fourreau=FilletBox(radius=1.7,point1=Oo,point2=Om)
CONTAINER = TechnologyEntity(    material=ALU_FOURREAU  ,
                                 localGeometry=Fourreau)

Eau_Fourreau=Cylinder(center=center,axis=vz,radius=3.4,height=60.0)
WATER     = TechnologyEntity(    material=EAU_FOU_BAR   ,
                                 localGeometry=Eau_Fourreau)

Barillet=Cylinder(center=center,axis=vz,radius=3.3,height=60.0)
BASKET    = TechnologyEntity(    material=ALU_BARILLET  ,
                                 localGeometry=Barillet)

#hole
hole = Cylinder(center = Oo, axis = vz,radius = 0.9, height = 60 )
HOLE_0    = TechnologyEntity(    material=EAU_BAR_CR      ,
                                 localGeometry=hole      )
HOLE_1    = TechnologyEntity(    material=EAU_BAR_CR      ,
```

```
                                        localGeometry=hole      )
HOLE_2    = TechnologyEntity(material=EAU_BAR_CR          ,
                                        localGeometry=hole      )
HOLE_3    = TechnologyEntity(material=EAU_BAR_CR          ,
                                        localGeometry=hole      )
HOLE_4    = TechnologyEntity(material=EAU_BAR_CR          ,
                                        localGeometry=hole      )
HOLE_5    = TechnologyEntity(material=EAU_BAR_CR          ,
                                        localGeometry=hole      )

#SURGAINE
surgaine=Cylinder(center=Oo,axis=vz,radius=0.74,height=60)
SURGAINE_0= TechnologyEntity(    material=SURGAINE        ,
                                        localGeometry=surgaine      )
SURGAINE_1= TechnologyEntity(    material=SURGAINE        ,
                                        localGeometry=surgaine      )
SURGAINE_2= TechnologyEntity(    material=SURGAINE        ,
                                        localGeometry=surgaine      )
SURGAINE_3= TechnologyEntity(    material=SURGAINE        ,
                                        localGeometry=surgaine      )
SURGAINE_4= TechnologyEntity(    material=SURGAINE        ,
                                        localGeometry=surgaine      )
SURGAINE_5= TechnologyEntity(    material=SURGAINE        ,
                                        localGeometry=surgaine      )

#GAINE
gaine=Cylinder(center=Oo,axis=vz,radius=0.27,height=60)
GAINE_0   = TechnologyEntity(    material=GAINE      ,
                                        localGeometry=gaine      )
GAINE_1   = TechnologyEntity(    material=GAINE      ,
                                        localGeometry=gaine      )
GAINE_2   = TechnologyEntity(    material=GAINE      ,
                                        localGeometry=gaine      )
GAINE_3   = TechnologyEntity(    material=GAINE      ,
                                        localGeometry=gaine      )
GAINE_4   = TechnologyEntity(    material=GAINE      ,
                                        localGeometry=gaine      )
GAINE_5   = TechnologyEntity(    material=GAINE      ,
                                        localGeometry=gaine      )

#rods
rod=Cylinder(center=Oo,axis=vz,radius=0.2699,height=60)
ROD_0     = TechnologyEntity(    material=UO2     ,
                                        localGeometry=rod       )
ROD_1     = TechnologyEntity(    material=UO2     ,
                                        localGeometry=rod       )
ROD_2     = TechnologyEntity(    material=UO2     ,
                                        localGeometry=rod       )
ROD_3     = TechnologyEntity(    material=UO2     ,
                                        localGeometry=rod       )
```

```
ROD_4      = TechnologyEntity(    material=UO2     ,
                                  localGeometry=rod      )
ROD_5      = TechnologyEntity(    material=UO2     ,
                                  localGeometry=rod      )

HOLE_0.transformation = translateRod
HOLE_1.transformation = Transformation(
        matrix_list = [translateRod,rotateRod])
HOLE_2.transformation = Transformation(
        matrix_list = HOLE_1.transformation.matrix_list+[rotateRod])
HOLE_3.transformation = Transformation(
        matrix_list = HOLE_2.transformation.matrix_list+[rotateRod])
HOLE_4.transformation = Transformation(
        matrix_list = HOLE_3.transformation.matrix_list+[rotateRod])
HOLE_5.transformation = Transformation(
        matrix_list = HOLE_4.transformation.matrix_list+[rotateRod])

CONTAINER.children = [WATER]
WATER.children   = [BASKET]
BASKET.children = [HOLE_0,HOLE_1,HOLE_2,HOLE_3,HOLE_4,HOLE_5]
for i in range(6) :
        cmd  = "HOLE_%d.children = [SURGAINE_%d]\n"%(i,i)
        cmd += "SURGAINE_%d.children = [GAINE_%d]\n"%(i,i)
        cmd += "GAINE_%d.children = [ROD_%d]"%(i,i)
        print cmd
        exec(cmd)
```

Annexe L

Validation numérique avec TRIPOLI4 du calcul neutronique présenté au chapitre 9

Les paramètres neutroniques validés par rapport à TRIPOLI4 sont :

- La réactivité du cœur ;
- La distribution des taux de fission par quart de plaque sur chaque plaque du cœur ;
- La fission sur chaque crayon (découpé en 5 couronnes concentriques) ;
- Le flux à 4 groupes sur les collectrons ;

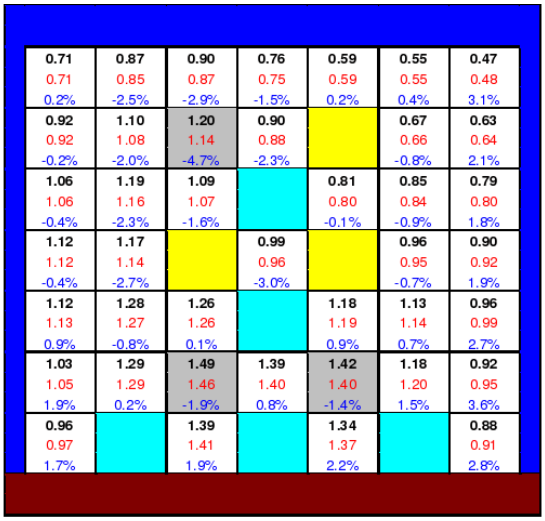


FIG. L.1 – Ecarts APOLLO2 − TRIPOLI4 sur les k_{eff} du calcul de cœur d'OSIRIS.

L'écart APOLLO2 − TRIPOLI4 sur la réactivité du cœur (figure L.1) est de −103 pcm (calculé en logarithme népérien), ce qui est très satisfaisant.

Les écarts sur la distribution de puissance par assemblage sont inférieurs à 4% sur les assemblages standard. En ce qui concerne les assemblages suiveurs, l'écart le plus important est situé sur la barre numéro 5 avec -4.7%, à un emplacement où le gradient de puissance est très important : il existe un écart de 33% entre la puissance sur cet assemblage et la puissance la plus faible d'un assemblage voisin (0.87).

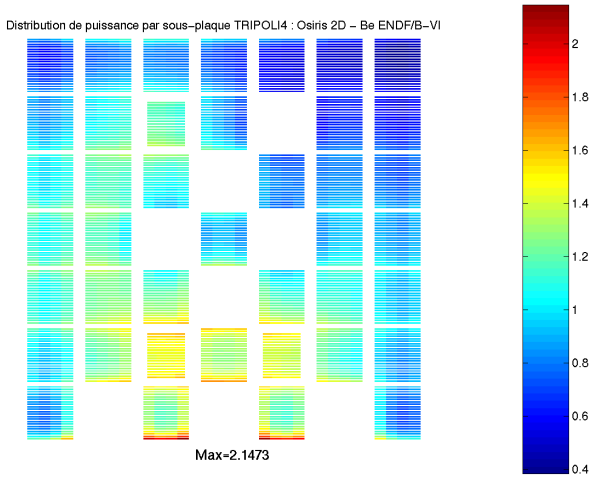


FIG. L.2 – Distribution de puissance par quart de plaque dans le cœur d'OSIRIS calculé par T4.

La distribution de puissance par $\frac{1}{4}$ de plaque (figure L.2) obtenue par le calcul TRIPOLI4 permet de visualiser le point chaud de ce calcul : la densité volumique de puissance est maximale sur les plaques connexes au réflecteur en béryllium sur les assemblages situés au centre de la première ligne d'assemblage, à cause du retour de neutrons fortement ralentis dans le béryllium. Dans les plans de chargement du réacteur, des assemblages usés sont généralement disposés à cet endroit pour limiter la remontée de la puissance.

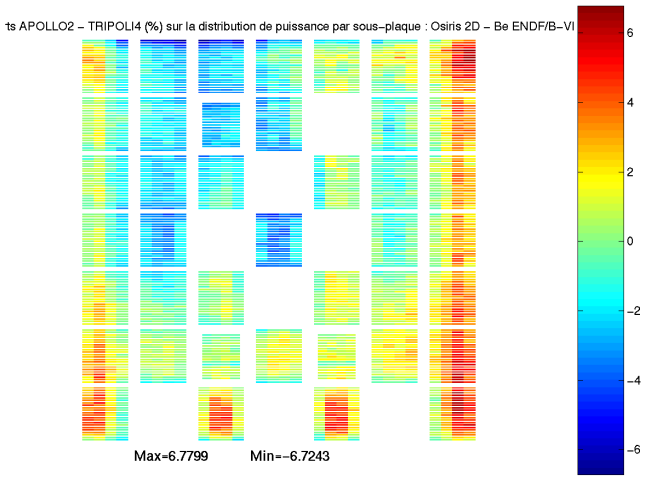


FIG. L.3 – Ecarts APOLLO2 – TRIPOLI4 sur les taux de fission par quart de plaque du cœur d'OSIRIS.

La répartition spatiale des écarts sur les taux de fission par plaque est représentée sur la figure L.3. Elle fait apparaître des écarts inférieurs en valeur absolue à 6.8%, et 95% des écarts compris entre -2.5% et $+2.5\%$ (figure L.4). L'incertitude statistique moyenne du calcul TRIPOLI4 à 2σ est égale à 0.9%, avec un écart maximal atteignant 1.4% (2σ) sur une plaque centrale de l'avant-dernier assemblage de la première rangée d'assemblages (facteur de puissance assemblage de 0.55). Compte-tenu des approximations réalisées pour ce calcul déterministe (calcul à nombre de groupes réduit, homogénéisation de la plaque) ces résultats sont très satisfaisants. Le temps de calcul est de 1 heure pour un pas d'évolution sur un PC linux 3GHz, et pourrait être réduit en optimisant le maillage spatial du cœur et du réflecteur.

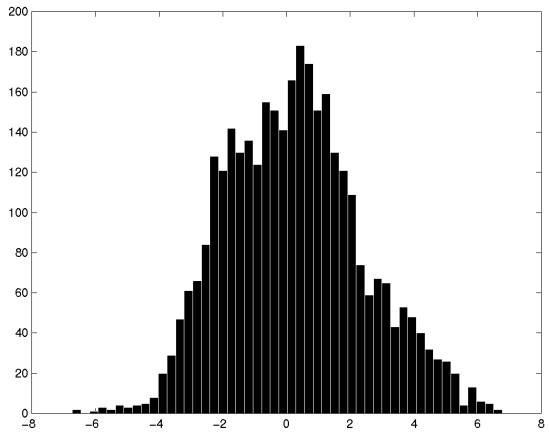


FIG. L.4 – Répartition des écarts de puissance par quart de plaque.

Les écarts sur la puissance linéique (W/cm) des crayons du dispositif TANOXOS (figure L.5) sont inférieurs à 4% en valeur absolue, même pour les crayons situés à 10 cm du cœur. La modélisation fine proposée dans cette étude permet de prendre en compte le gradient radial de puissance sur les crayons lié au mode de pilotage du réacteur OSIRIS (de l'ordre de 4% avec cette position des barres de contrôle). On peut ainsi obtenir des facteurs ($\frac{P}{i}$) prenant en compte cette hétérogénéité radiale. Ces écarts se retrouvent également sur les flux thermiques (< 0.625eV) calculés sur les collectrons (SPND[1]).

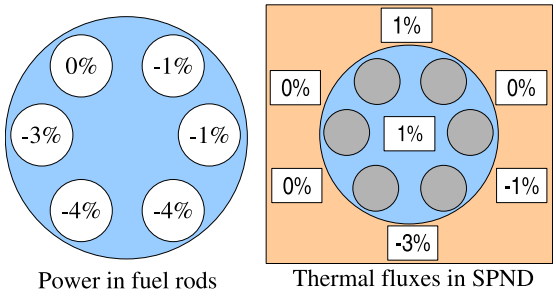


FIG. L.5 – Comparaison TRIPOLI4/APOLLO2 (tirée de [BSA+06]) des puissances calculées dans les crayons et des flux thermiques des collectrons de TANOXOS.

[1] SPND : Self Power Neutron Detector

Annexe M

Importation de fichier STEP dans SILENE

Dans le paragraphe 9.2.1 sont détaillées les solutions techniques adoptées pour importer des géométries décrites dans des fichier au format STEP (ISO 10303 AP 214) dans le modèle de données interne de SILENE. Cette annexe présente l'utilisation des développements réalisés dans SILENE.

M.1 Fonctionnalités ajoutées

SILENE prend en charge des géométries 2D planes ou des géométries 3D provenant de géométries 2D extrudées. Dans le cadre des développements nous ne nous sommes intéressés qu'aux géométries 2D planes. De ce fait, les fonctionnalités STEP ajoutées à SILENE ne traitent que des géométries 2D planes sur l'axe des z.

L'ouverture d'un fichier au format STEP a été ajoutée dans le menu Silene (figure M.1).

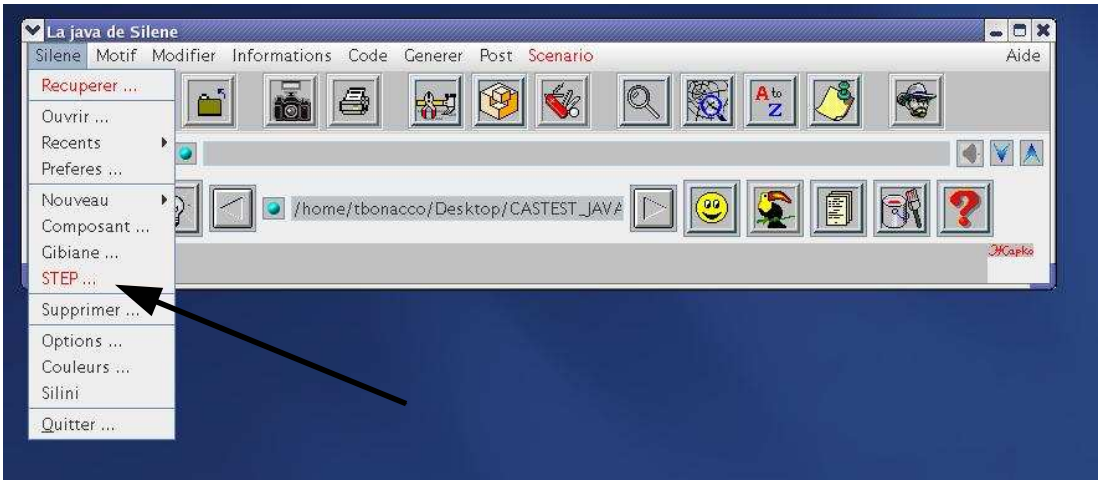


FIG. M.1 – Modification de l'interface SILENE

Une interface permet de choisir le fichier STEP à ouvrir (figure M.2).

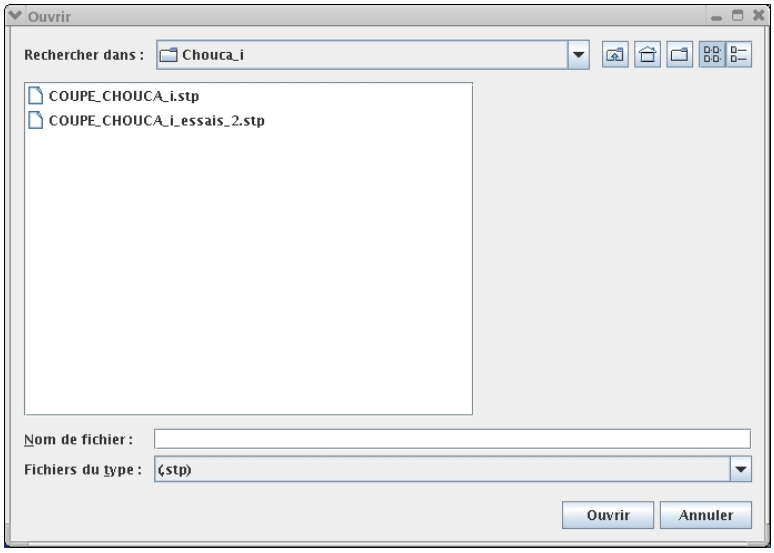


FIG. M.2 – Choix du fichier STEP

La géométrie STEP ne contient pas d'information sur le nom des milieux, il est possible d'associer un fichier renseignant sur le nom de chacun des milieux (figure M.3 et figure M.4). A chaque maille est associé un milieu différent. Par défaut aucun fichier de milieux n'est associé. Les milieux créés pour chaque maille sont alors nommés `mil\_x`, où `x` est un nombre.

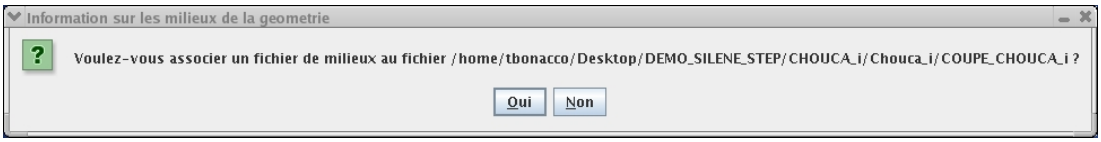


FIG. M.3 – Interface proposant d'associer un fichier de milieux

Nous proposons le format de fichier de milieux suivant :

Un fichier ASCII d'extension *.mil* . La première ligne renseigne sur le nombre de milieux que l'on peut lire dans le fichier. Ensuite chaque ligne renseigne sur le nom court du $(n-1)^{ieme}$ milieu (où n correspond au numéro de ligne), suivi du caractère tabulation, suivi du nom long. Un exemple de fichier est donné par le tableau M.1. Dans notre étude, ce fichier est créé par script PYTHON, à partir d'un modèle PYTHON qui décrit la géométrie d'un objet.

Les milieux sont numérotés dans l'ordre dans lequel les instances Advanced_face apparaissent dans le fichier STEP.

La géométrie du fichier STEP est convertie dans la représentation de SILENE (figure M.5). Celle-ci peut ensuite être maillée, enregistrée au format SILENE standard *.dat* et il est possible de générer le fichier au format *TDT*, nécessaire au calcul MOC du code

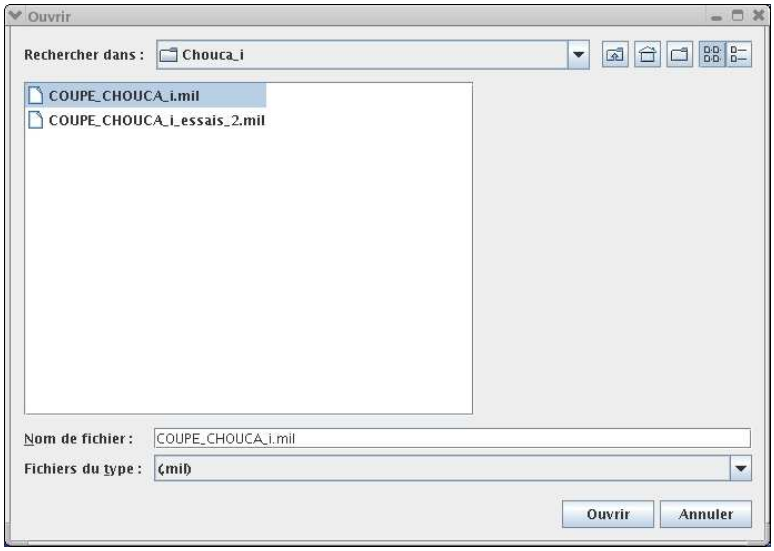


FIG. M.4 – Choix du fichier de milieux

```
6
MIL_ECHA_CHOUCA MILIEU_ECHA_CHOUCA
MIL_INOX_CHOUCA_INT MILIEU_INOX_CHOUCA_INT
MIL_FOUR_CHOUCA MILIEU_FOUR_CHOUCA
MIL_ZRO2_CHOUCA MILIEU_ZRO2_CHOUCA
MIL_INOX_CHOUCA_EXT MILIEU_INOX_CHOUCA_EXT
MIL_EAU_CHOUCA_EXT MILIEU_EAU_CHOUCA_EXT
```

TAB. M.1 – Exemple de fichier de milieux

APOLLO2.

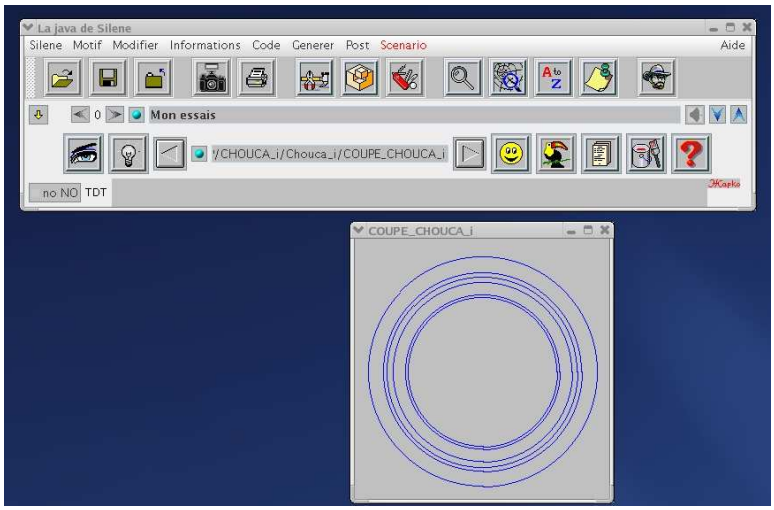


FIG. M.5 – Ouverture d'une géométrie STEP dans SILENE

M.2 Utilisation dans SALOME

M.2.1 Echange d'une géométrie du composant GEOM vers SILENE

Le but de cet exercice est de créer une géométrie avec le composant GEOM, de l'exporter au format STEP, et de l'ouvrir dans SILENE. Nous commençons par travailler sur un cas d'école représentant une géométrie 2D en croissant de lune qui contient un carré (figure M.6). L'intérêt de ce cas est qu'il contient d'une part la notion d'objet contenu et d'objet contenant, d'autre part il possède des bords en arcs de cercles et des bords droits.

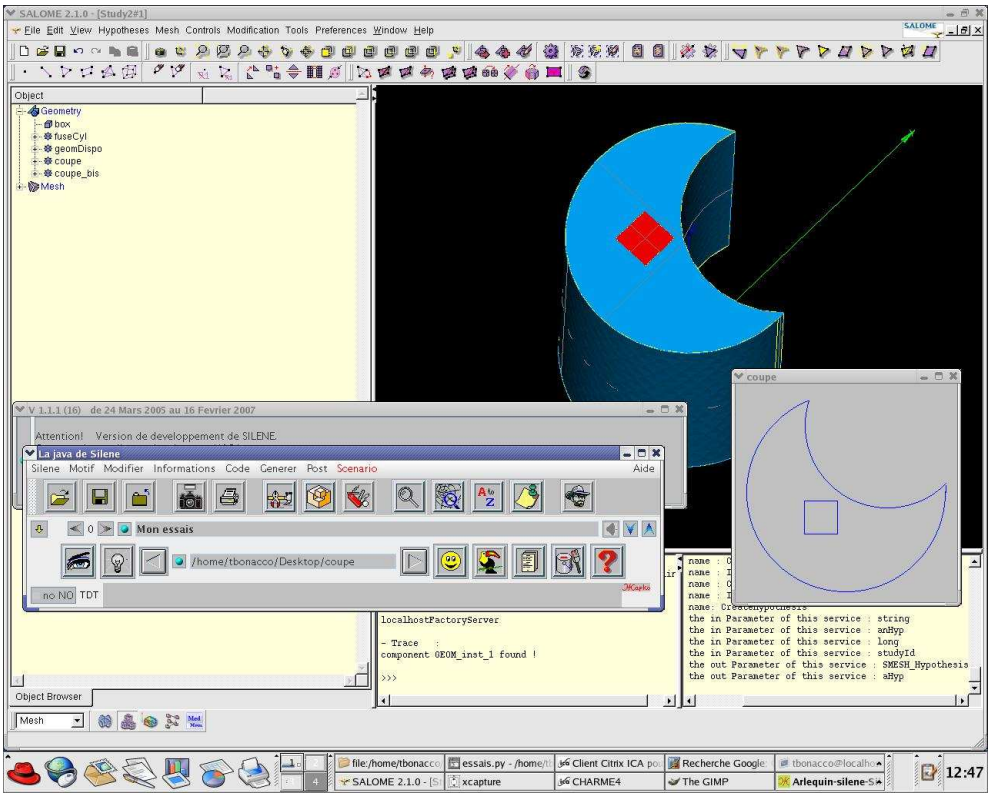


FIG. M.6 – Ouverture du cas école dans SILENE

La géométrie créée à l'aide de SALOME peut être réalisée soit en utilisant les outils de l'interface graphique, soit par script PYTHON. A l'origine nous construisons chacun des objets en 3D. Dans le cas école nous avons deux objets : le croissant de lune (auquel ont été retirés les objets qu'il contient) construit par la différence de deux cylindres et un parallélépipède (dont la section est un carré). Ensuite nous définissons une composition d'objets à partir de chacun des objets 3D. Enfin nous réalisons une coupe de la composition suivant l'axe z.

La coupe de la géométrie ainsi créée est alors exportée au format STEP par le composant GEOM de SALOME (File→Export...). Le fichier STEP peut être ouvert dans SILENE (figure M.6)).

M.2.2 Application à un dispositif expérimental complexe : TANOXOS

Nous ne détaillerons pas ici le script ou le modèle d'Entités Technologiques permettant de construire la géométrie.

Les objets géométriques de chacun des milieux proviennent de fichiers STEP créés auparavant. Cet exercice, présenté par la figure M.7, valide l'importation de la géométrie STEP du dispositif complet dans SILENE, ainsi que l'importation des milieux associés par le fichier de milieux.

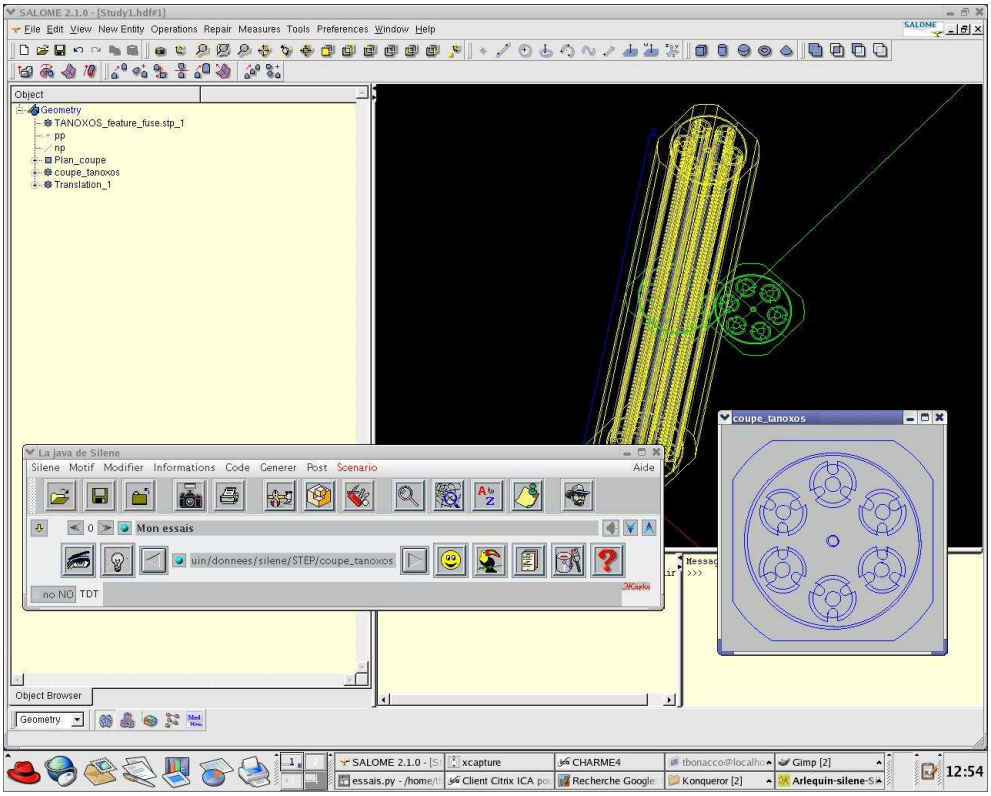


FIG. M.7 – Application au dispositif TANOXOS

Par ailleurs, un outil (une méthode de la classe `TechnologyEntity`) a été mis en œuvre pour traduire un modèle d'Entités Technologiques 3D en une représentation 2D sous la forme d'un couple de fichiers *.step* et *.mil* convenablement formaté.