Couplage et synchronisation de modèles dans un code scénario d'accidents graves dans les réacteurs nucléaires

Louis Viot

université PARIS-SACLAY

école normale supérieure paris-saclay

NNT : 2018SACLN033

Thèse de doctorat

Couplage et synchronisation de modèles dans un code scénario d'accidents graves dans les réacteurs nucléaires

Thèse de doctorat de l'Université Paris-Saclay
préparée à l'Ecole normale supérieure de Cachan

École doctorale n°574
École doctorale de mathématiques Hadamard (EDMH)
Spécialité de doctorat : Mathématiques appliquées

Thèse présentée et soutenue à Cachan, le 12 octobre 2018, par

Louis Viot

Composition du Jury :

Marc MÉDALE IUSTI UMR CNRS 6595	Président du jury
Florent DUCHAINE CERFACS	Rapporteur
Frédéric NATAF Université Paris 6	Rapporteur
Laure QUIVY CMLA UMR CNRS 8536	Examinateur
Florian DE VUYST LMAC EA 222	Directeur de thèse
Laurent SAAS CEA Cadarache	Encadrant de thèse

In the Land of Mordor where the Shadows lie.
One Ring to rule them all, One Ring to find them,
One Ring to bring them all, and in the darkness bind them,
In the Land of Mordor where the Shadows lie.

The Fellowship of the Ring,
J.R.R. Tolkien.

In the Land of `PROCOR` *where the Models lie.*
One Platform to rule them all, One Platform to find them,
One Platform to bring them all, and in the darkness bind them,
In the Land of `PROCOR` *where the Models lie.*

The author.

Résumé

Le travail de thèse détaillé dans ce manuscrit présente la résolution numérique *partitionnée* et *synchronisée* des *systèmes complexes* survenant lors de la simulation d'accidents graves dans les réacteurs à eau légère au sein de la plate-forme logicielle CEA PROCOR.

La simulation d'accidents graves se fait par la résolution de systèmes complexes constitués des différents phénomènes physiques couplés apparaissant dans le réacteur nucléaire. Les phénomènes sont multi-physiques, par exemple la neutronique, la thermohydraulique, la thermochimie, la mécanique, etc., et multi-échelles avec des temps caractéristiques allant de la microseconde à l'année, et des masses allant du gramme à la centaine de tonnes. De plus, le manque de données et de connaissances phénoménologiques font que la modélisation des différents phénomènes, encore incertaine, est en constante évolution. Enfin, les modèles à états du système peuvent déclencher des événements de changement d'état associés à des discontinuités.

Permettant de traiter chaque modèle avec une résolution numérique adaptée au phénomène sous-jacent et offrant modularité et évolutivité des modèles, l'approche partitionnée de résolution des systèmes complexes est par conséquent particulièrement adaptée au contexte des accidents graves. Cette approche propose des algorithmes partitionnés de résolution des problèmes couplés associés aux systèmes complexes permettant, en faisant le bon choix de l'algorithme, de retrouver la précision d'une résolution monolithique totalement couplée.

Dans ce travail de thèse, un formalisme de représentation des systèmes complexes a été proposé. Les algorithmes partitionnés ont été étudiés et adaptés pour la résolution de ces systèmes. Un algorithme de synchronisation sur les événements des modèles du système a été proposé. Une architecture logicielle de couplage, dans laquelle ont été intégrés les différents algorithmes de partitionnement et de synchronisation, a été ajoutée à la plate-forme PROCOR. Des résultats numériques sur des couplages d'importances pour les accidents graves et sur des cas industriels valident et montrent le potentiel des algorithmes implémentés. Cette approche et les résultats associés sont généraux et pourraient s'appliquer à d'autres domaines.

Abstract

This PhD topic is focused on the *partitioned* and *synchronised* numerical solving of *complex systems* arising in the simulation of severe accidents in light water reactors by the CEA PROCOR platform.

The simulation of severe accidents is made possible by the solving of complex systems composed of the various coupled physical phenomena appearing inside the nuclear reactor. Phenomena are multi physics, such as neutronic, thermal hydraulic, thermochemistry, mechanic, etc., and multi scales, with characteristic times going from microseconds to years and masses going from grams to hundred of tons. Furthermore, because of the lack of physical data and phenomenological knowledge the modeling of the various phenomena is still uncertain and in constant development. Finally, models of the system can trigger event corresponding to internal change of state which are often associated with strong discontinuities.

Allowing to handle each model with a numerical scheme adapted to the underlying physical phenomenon and offering modularity and evolutivity, the partitioned approach of solving the complex systems is particularly suitable in the context of severe accidents. This approach offers many partitioned algorithms to solve the coupled problems associated to the complex systems and, with the appropriate choice of algorithm, the precision of the monolithic solving can even be retrieved.

In this work, a new formalism to represent the complex systems is described. The partitioned algorithms are studied and adapted to the solving of these systems. A new synchronization algorithm is proposed and allow the coupled models to be synchronised on their internal events. Furthermore, an coupling software architecture, in which the previous algorithms have been implemented, has been added to already existing PROCOR industrial platform. Numerical results on important couplings for severe accidents and from industrial applications of the PROCOR software are given and allow to validate the implemented algorithm and show the promising potential of the coupling environment.

Remerciements

Ma plus profonde gratitude à mon encadrant Laurent Saas d'avoir su trouver du temps pour moi dans son emploi du temps (*très*) chargé. Merci infiniment pour la confiance que tu m'as accordée durant la thèse et que tu continues à me témoigner. J'espère pouvoir, dans un futur (*très*) proche, te rendre la monnaie de ta pièce.

Un grand merci à Florian De Vuyst, mon directeur de thèse, pour ses idées, son enthousiasme perpétuel, son optimisme et ses encouragements. Vous m'avez permis, entre autres, d'avoir un aperçu du monde de la recherche universitaire et de m'y immiscer brièvement. Enfin, il serait *ingrat* de ma part de ne pas vous remercier de m'avoir fait découvrir les mets et spécialités culinaires asiatiques de Cachan et Compiègne dont je suis si friand.

Ma sincère reconnaissance à Florent Duchaine et Frédéric Nataf d'avoir accepté d'être les rapporteurs de ma thèse et pour l'intérêt que vous y avez porté. Un grand merci également à Laure Quivy et Marc Médale d'avoir participé au jury de ma thèse et pour les questions pertinentes posées durant ma soutenance.

Discussions scientifiques, longues pauses café et repas interminables à la cantine ont rythmé ma vie de thèse. Pour tous ces bons moments, je remercie tous mes collègues du CEA. En particulier, Benoît Habert (et son insatiable appétit) d'avoir su me rassurer tout au long de la thèse avec sa perpétuelle bonne humeur, Romain Le Tellier pour ses bons conseils et ineffables connaissances scientifiques et cinématographiques, Mathieu Peybernes pour son recul sur la vie (et les voyages en 106), et Rémi Clavier pour son inattendue relecture complète de mon manuscrit. Un grand merci également à Matthieu Grivelet de m'avoir supporté et enseigné les arts de l'escalade et de la vie provençale (puis de m'avoir abandonné) et à Patrick Ah-Fat pour tout, tout simplement. Enfin, une inexprimable reconnaissance à Gabriel Gay pour sa mansuétude envers mes humeurs tourmentées et instables, sa profonde compréhension de ma personne, sa bonne humeur quotidienne, et pour m'avoir fait découvrir des horizons culinaires que, jamais, Ô grand jamais, je n'eusse osé imaginer.

Une sincère, grande et profonde reconnaissance à l'ensemble de ma famille pour avoir supporter mes réactions *aléatoires*, mon comportement *incer-*

tain et mes décisions *chaotiques* m'amenant (*trop*) souvent à faire la visite des plus beaux hôpitaux de France. Pour toute l'attention que vous m'avez apportée, quelque jour, je vous le revaudrai, si je puis, car comme Rimbaud, « *Je ne pourrai jamais envoyer l'Amour par la fenêtre* » (*Phrases, Illuminations*).

À celles et ceux que j'ai pu oublier ou à celles et ceux avec qui je n'aurai plus l'occasion de travailler mais à celles et ceux qui ont, néanmoins, participé à ma construction scientifique et personnelle tout au long de la thèse, *merci !* Et si par malheur nous n'avons plus l'occasion d'être de nouveau réunis, si ma présence vient un jour à vous manquer ou mon absence vous attrister, sachez que « *tout chagrin passe. Tout bonheur perdu n'est que l'attente d'un bonheur plus grand.* » (*La hache et la croix, François Cavanna*).

Table des matières

Table des figures

Liste des tableaux

Glossaire

AG Accident Grave. 1, 2, 8–17, 19, 23, 25, 26, 29, 30, 32, 35, 42, 52, 63, 65, 66, 68, 107, 109, 148, 149, 155, 179, 182–185, 187, 193

APRP Accident de Perte de Réfrigérant Primaire. 9, 10

CFD Computational Fluid Dynamics. 13

CPS Conventionnal Parallel Staggered. 45, 46

CSS Conventionnal Serial Staggered. 45–47, 50, 99, 100, 102, 184

EA Équation Algébrique. 73–75, 79, 186

EDA Équation Différentielle Algébrique. 73, 74, 186

EDO Équation Différentielle Ordinaire. 72–75, 79, 186, 195

EPR Evolutionary Power Reactor. 6

Focusing Effect Phénomène caractérisant une dissymétrie entre la puissance évacuée par transfert radiatif par la surface supérieure et par conduction par la surface latérale de la couche d'acier liquide au dessus d'un bain de corium en fond de cuve : lors du "focusing effect", la majeure partie de la puissance est évacuée par la surface latérale car le transfert radiatif est beaucoup moins efficace, créant, lors d'une couche mince d'acier, une concentration du flux de chaleur sur cette surface. Ce phénomène est d'autant plus marqué lorsque la couche d'acier est fine et peut mener au percement de la cuve si le flux de chaleur imposé est trop important. 19, 21, 150, 151, 155, 167, 170

GSS Generalized Serial Staggered. 45–47

IPS Improved Parallel Staggered. 45

ISS Improved Serial Staggered. 45

IVR In Vessel Retention. 15, 16, 19, 163, 167, 175

LBLOCA Large Break Loss-Of-Coolant Accident. 9

LOCA Loss-Of-Coolant Accident. 9

LOOP Loss Of Off Site Power. 9, 154, 155, 179

LP Lumped Parameter. 107, 190, 191, 193, 195

PSA Probability Safety Analysis. 14, 155, 187

REP Réacteur à Eau Pressurisée. 5, 7–9, 11, 13

SBLOCA Small Break Loss-Of-Coolant Accident. 9, 179

SCE Schéma de Couplage Explicite. 41, 42, 44–48, 50, 55, 57, 67, 68, 93, 99, 108, 136, 138, 140–145, 148, 159, 161, 188

SCI Schéma de Couplage Implicite. 41, 42, 44, 48–51, 53, 55, 57–61, 67, 68, 93, 99, 104, 108, 136–138, 144–146, 156, 175, 179, 180, 182, 186, 188

Introduction générale

Le travail de thèse détaillé dans ce manuscrit présente la résolution numérique *partitionnée* et *synchronisée* des *systèmes complexes* survenant lors de la simulation d'accidents graves (AG) dans les réacteurs à eau légère au sein de la plate-forme logicielle PROCOR. Un tel accident met en jeu de nombreux phénomènes multi-physiques et multi-échelles tels que la neutronique, la thermohydraulique, la thermochimie, la mécanique. Ces phénomènes sont couplés entre eux avec des temps caractéristiques allant de la seconde à l'année, et des masses allant du gramme à plusieurs tonnes. La difficulté de créer des expériences exploitables, du fait des conditions extrêmes de température ($T \geq 3000$ K) et de la mise à l'échelle, et l'extraction et l'analyse parfois complexe de données souvent post-mortem des précédents AG font que la modélisation des phénomènes couplés et la phénoménologie de l'AG sont incertaines et amenées à fortement *évoluer*.

C'est, en complément des programmes expérimentaux, les calculs fournis par les plate-formes AG qui permettent l'amélioration des connaissances phénoménologiques et de modélisation des AG. Ces plate-formes sont utilisés dans deux cadres différents. D'une part, dans un cadre industriel pour des calculs statistiques sur l'ensemble ou une partie du scénario de l'AG, nécessitant une modélisation moins fine et une résolution grossière du couplage entre les phénomènes permettant des calculs rapides. D'autre part, elles sont utilisées pour des études précises concentrées sur le couplage entre quelques phénomènes, nécessitant une modélisation fine de ces phénomènes et une résolution précise du couplage entre ces phénomènes. Néanmoins, pour toutes ces utilisations de la plate-forme, le couplage entre phénomènes est *fixe* et a priori bien *connu* au contraire des modèles couplés qui peuvent être très *différents*, amenés à fortement *évolués* et être *réutilisés* plusieurs fois dans différents contextes d'utilisation.

Pour ces raisons, les systèmes complexes sont découpés en *blocs* couplés, représentant une partie géométrique du réacteur ou un phénomène physique apparaissant lors de l'AG, facilement échangeables et qui peuvent être, espérons le, rassemblés par la suite de manière à obtenir une résolution par blocs aussi proche que possible de la résolution *monolithique* en un seul bloc totale-

ment couplé. Cette approche de résolution du couplage des modèles correspond à l'approche *partitionnée*. Dans cette approche, les modèles couplés sont vus comme des *boîtes noires* avec un ensemble d'entrées et de sorties. Ainsi, elle offre la *modularité*, l'*encapsulation* et la possibilité de *réutilisation de codes* qui sont imposées par le contexte AG. Cependant, une résolution "naïve" du partitionnement est connue pour être moins stable et moins précise que la résolution monolithique des problèmes couplés.

De plus, dans la modélisation des AG, les modèles constituant le système complexe sont généralement des modèles à événements. Chaque événement correspond à un changement d'état du modèle, par exemple passage de l'état liquide à l'état solide, et donc à un changement des équations physiques le décrivant. Par conséquent, le système complexe est amené à muter au cours de la simulation et, au moment de sa mutation, des discontinuités peuvent apparaître dans les fonctions mathématiques le définissant. D'une part parce qu'une condition sine qua non d'application de la plupart des théorèmes mathématiques, par exemple la convergence des itérations de Newton, est la continuité des fonctions et d'autre part parce que ces événements sont parfois d'intérêt pour la simulation, par exemple le percement d'une cuve de réacteur, la résolution partitionnée doit pouvoir se *synchroniser* sur les différents événements des modèles du système complexe.

Les efforts initiaux de développement dans la plate-forme PROCOR ne se sont pas portés sur l'aspect logiciel et numérique du couplage de modèle. La résolution du partitionnement se fait par simple chaînage des modèles connue pour être *peu précise*, *parfois instable* et n'offrant des possibilités de synchronisation que *très limitées*. Les objectifs de cette thèse sont multiples :

1. Définir un formalisme permettant de représenter les systèmes complexes apparaissant lors de la simulation des AG.
2. Étudier et adapter les algorithmes partitionnés permettant la résolution des problèmes couplés définis par ces systèmes complexes.
3. Adapter les algorithmes partitionnés pour permettre la synchronisation sur les événements déclenchés par les modèles couplés.
4. Créer et intégrer dans la plate-forme PROCOR une architecture de couplage de modèles offrant la modularité et l'évolutivité offerte par l'approche partitionnée et permettant l'utilisation des algorithmes de partitionnement et de synchronisation.

Le présent manuscrit est composé de 4 chapitres.

Dans le chapitre 1 sont présentés le contexte physique et le contexte logiciel des accidents graves dans lesquels ce travail de recherche s'inscrit.

Le chapitre 2 est consacré à la définition d'un formalisme permettant la représentation des systèmes complexes ainsi qu'à la présentation des différents

algorithmes de résolution du partitionnement et de l'algorithme de synchronisation.

Le chapitre 3 détaille l'architecture logicielle d'avant thèse de la plate-forme PROCOR et l'environnement logiciel de couplage introduit dans celle-ci dans le cadre de la thèse.

Le chapitre 4 présente au travers de résultats numériques sur des cas d'études et des cas industriels l'utilisation des algorithmes de partitionnement et de synchronisation dans la nouvelle plate-forme PROCOR.

Enfin, des conclusions sur le travail effectué dans cette thèse et des ouvertures possibles sont données.

Chapitre 1

Contexte

Ce chapitre permet d'introduire le contexte physique des accidents graves dans les réacteurs nucléaires dans lequel cette thèse s'inscrit. Dans la section 1.1, on introduira des éléments permettant de comprendre :

- Le fonctionnement de certains réacteurs nucléaires.
- La phénoménologie des accidents pouvant survenir dans ces réacteurs.
- La modélisation de ces accidents.
- Les différents types de codes et logiciels permettant la simulation de ces accidents.

Le contexte logiciel et numérique de la thèse sera détaillé dans la section 1.2. Enfin, les objectifs de la thèse seront donnés dans la section 1.3.

1.1 Les accidents graves dans les réacteurs à eau pressurisée

L'objectif principal de la sûreté nucléaire est de minimiser les risques que présentent un ou plusieurs réacteurs nucléaires sur la population et sur l'environnement, proche ou à distance, et la sécurité du personnel gérant les installations nucléaires. Ces risques sont liés à la dispersion de produits radioactifs survenant à la suite d'un accident au sein du réacteur. Nous ne parlerons par la suite que des réacteurs français à eau pressurisée (REP). Les autres réacteurs tels que les réacteurs bouillants de Fukushima Daiichi, graphites de Tchernobyl ou à neutrons rapides (type ASTRID) ne seront pas traités. Néanmoins, l'extension à ces types de réacteurs est possible en prenant en compte la géométrie et les phénomènes physiques propres à ces réacteurs.

1.1.1 Introduction au fonctionnement et à la sûreté dans les REP

Au total, 58 réacteurs nucléaires électrogènes à eau pressurisée sont présents sur le sol français. 34 d'entre eux fournissent une puissance de 900 MWe, 20 réacteurs fournissent une puissance de 1300 MWe et enfin 4 réacteurs four-

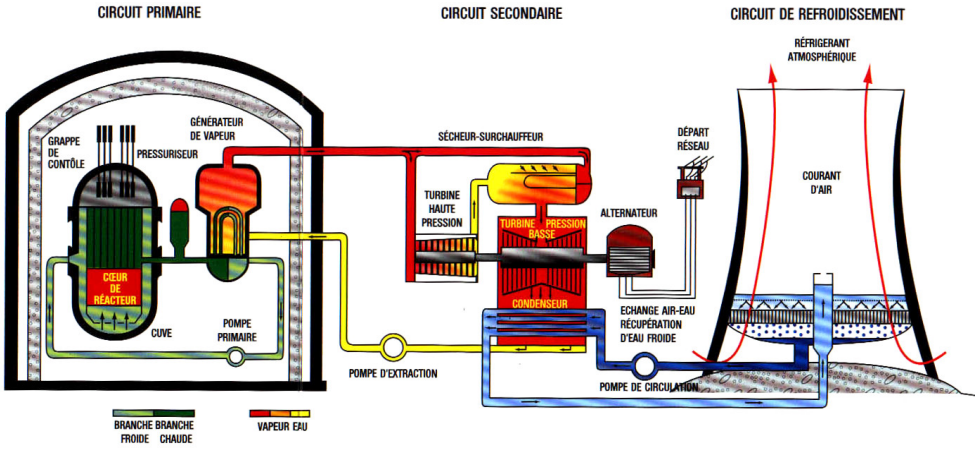


FIGURE 1.1 – Principe de fonctionnement d'un réacteur à eau pressurisée

nissent une puissance de 1450 MWe [54]. Le nouveau réacteur "Evolutionary Power Reactor" ou "European Pressurised Reactor" (EPR) est en construction à Flamanville. Sa puissance s'élève à 1650 MWe. En 2015, d'après le bilan réalisé par le réseau de transport d'électricité de France, 76, 3% de l'énergie produite en France était issue des réacteurs nucléaires en fonctionnement.

Dans les réacteurs de puissance, cette énergie provient des réactions neutroniques du cœur du réacteur qui est extraite à l'aide d'un fluide calo-porteur, l'eau, sous forme de chaleur. La transformation de l'énergie chaleur en énergie électrique se fait en plusieurs étapes au sein du réacteur (voir figure 1.1) :

1. L'eau circule en circuit fermé dans le circuit primaire du réacteur à la pression de 155 bars et à une température pouvant atteindre 320°C et extrait la chaleur du cœur contenant le combustible en le refroidissant.
2. La chaleur récupérée par l'eau dans le circuit primaire est transmise à l'eau du circuit secondaire au niveau du générateur de vapeur. Étant à une pression inférieure, elle s'évapore et alimente le groupe turbo-alternateur produisant l'électricité.
3. Le restant de l'énergie contenue dans l'eau est ensuite transmise à l'extérieure.

La chaleur extraite du cœur, moteur du réacteur nucléaire, provient principalement des réactions de fission. Elles agissent sur les noyaux de certains isotopes tels que l'^{235}U contenus dans le combustible nucléaire disposé dans le cœur du réacteur dans des assemblages de crayons de combustibles (voir 1.2). Lors d'une réaction de fission, sous l'impact de neutrons, le noyau des isotopes est scindé en deux et des fragments en mouvement apparaissent. La chaleur est extraite de l'énergie cinétique de ces fragments. La réaction de fission des isotopes peut être induite volontairement par un jet de neutrons ou

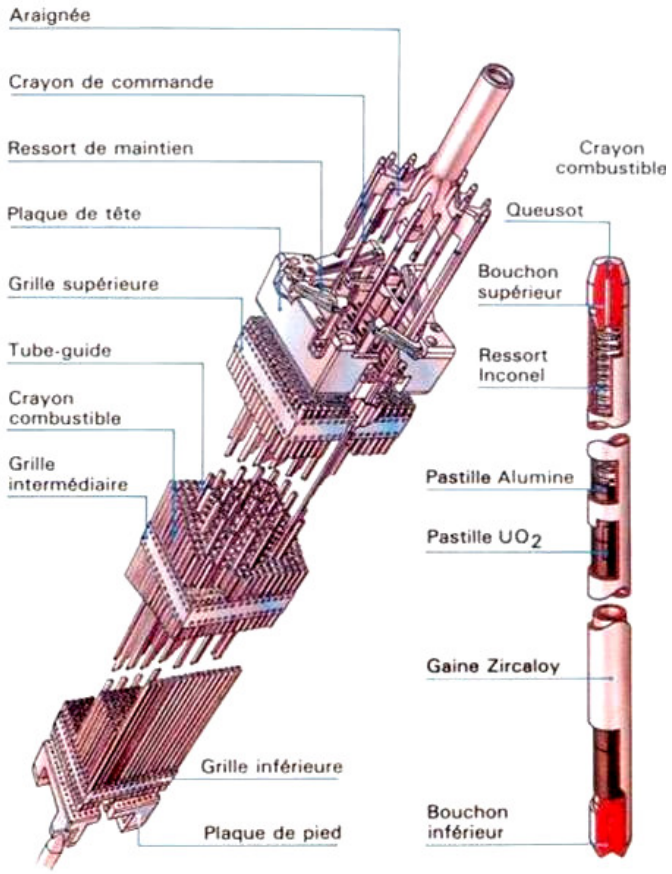


FIGURE 1.2 – Assemblage combustible (gauche) et crayon combustible (droite)

est spontanée dans le cas d'isotopes plus lourds. La fission d'un isotope est accompagnée du relâchement de un ou plusieurs neutrons provoquant des réactions de fission en chaîne. Pour que la réaction soit stable, il faut que le nombre de neutrons qui apparaissent soit strictement égal à celui de ceux qui disparaissent, le système est alors dit *critique*. Si l'on est au dessus de la criticité, en sur-criticité, la réaction s'emballe et le nombre de produits de fission croît exponentiellement. L'objectif pour une production stable d'énergie électrique par le réacteur nucléaire va être d'opérer celui ci *juste* au dessus de la criticité. Le sur-plus de neutrons produit étant des neutrons plus lents relâchés par les réactions de fission, la neutronique du cœur est plus lente, plus stable et auto-régulatrice. Il est possible d'amener le réacteur dans cette zone cible juste au dessus de la criticité par le biais de barres de commande en cadmium ou bore, ou à l'aide de bore dissous dans l'eau.

La chaleur extraite du cœur peut aussi provenir d'une puissance résiduelle due à la désintégration radioactive des produits de fission. Cette puissance résiduelle décroît en fonction du temps et représente, une heure après l'arrêt du réacteur, environ 1.5% de sa puissance de fonctionnement nominale.

Ainsi, pour assurer la sûreté d'un REP à tout moment de son fonctionnement, il est nécessaire de **maîtriser** la réactivité du cœur pour assurer l'**extraction de l'énergie** émise sous forme de chaleur tout en **assurant le confinement** des matières radioactives. Ce confinement des substances est assurée par trois barrières physiques successives présentes dans le réacteur : la gaine des crayons combustibles, le circuit primaire fermé (la cuve du réacteur) et l'enceinte de confinement en béton du réacteur. Pour pallier les défaillances humaines ou physiques, par exemple la rupture de gaines, la fuite du circuit primaire ou de l'enceinte de confinement du réacteur, a été introduite la stratégie de défense en profondeur [88], divisée en plusieurs niveaux. A chaque niveau, correspondant à un état de fonctionnement du réacteur, est associé des dispositions prévenant l'évolution du réacteur vers une situation plus grave.

A ce concept de défense en profondeur purement déterministe, issu de retours d'expérience et de recherches, est associée une approche de sûreté probabiliste. Elle trouve tout son intérêt dans les réacteurs nucléaires du fait du grand nombre de défaillances physiques et humaines possibles.

Ces deux approches, déterministe et probabiliste, permettent de définir un ensemble de mesures censées prévenir et corriger tout incident pouvant survenir dans un réacteur. Ainsi, l'extraction de l'énergie émise par le cœur, la maîtrise de la réactivité du cœur et l'intégrité des barrières physiques sont assurés prévenant le rejet de déchets radioactifs dans l'environnement.

Malgré toutes ces mesures, une accumulation d'incidents, issus d'erreurs matérielles ou humaines, peut amener le réacteur dans une situation au cours de laquelle le combustible est dégradé par une fusion partielle ou complète du cœur : on parle alors d'**accidents graves** (AG).

1.1.2 Phénoménologie d'un accident grave dans un REP

Les AG les plus tristement célèbres sont ceux de *Three Miles Island* (TMI) en 1979, qui a entraîné la fusion partielle du cœur du réacteur (voir figure 1.3) avec peu de produits radioactifs relâchés dans l'environnement, celui de *Chernobyl* en 1986, ou celui de *Fukushima Daiichi* en 2011. De ces accidents peuvent être tirés des enseignements, parfois long à obtenir (plus de sept ans pour l'accident de *Three Miles Island*), permettant d'améliorer les connaissances phénoménologiques des AG pour ainsi assurer la sûreté du parc nucléaire mondial. L'accident de TMI est le plus riche d'enseignement pour la filière REP bien que seules des données post-mortem ont pu êtres extraites. L'accident

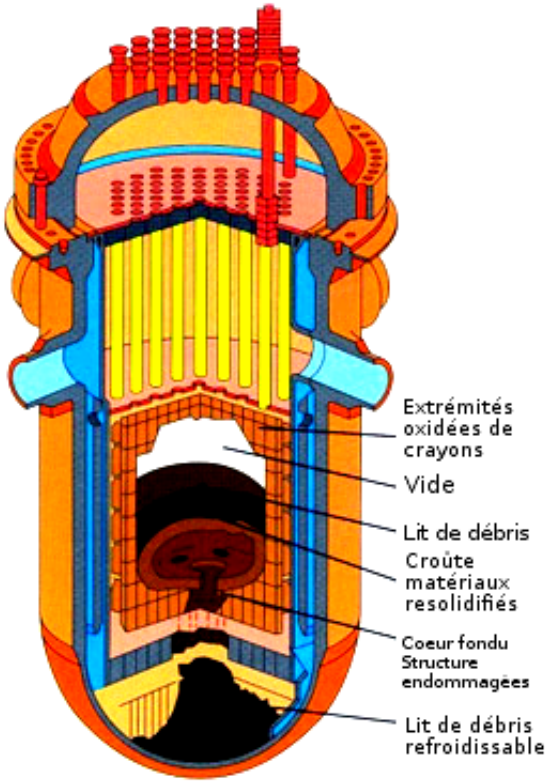


FIGURE 1.3 – Coupe de la coeur du réacteur de Three Miles Island après fusion partielle du celui ci (extrait de http://www.irsn.fr).

de Fukushima est encore trop récent et celui de Chernobyl concerne une filière différente avec une phénoménologie de l'accident trop éloignée.

Un *initiateur* d'AG peut par exemple être une insertion trop importante de réactivité dans le cœur du réacteur par le retrait des barres de commandes permettant de réguler sa neutronique. Ces accidents sont des *accidents de réactivité*. Ils ne seront pas traités dans ce manuscrit.

Dans ce manuscrit, on ne considère que les accidents graves initiés par la perte de l'eau, le réfrigérant primaire. On parle alors d'accidents de perte du réfrigérant primaire (APRP) ou "Loss Of Coolant Accident" (LOCA). La perte du réfrigérant peut être causée par une brèche de plus ou moins grande taille dans le circuit primaire du réacteur, auquel cas on parle de "Large Break Loss Of Coolant Accident" (LBLOCA, début de l'AG en 1 ou 2 h) ou de "Small Break Loss Of Coolant Accident" (SBLOCA, début de l'AG en 4 ou 6 heures). La perte du réfrigérant peut aussi être due à une perte de l'électricité permettant, entre autres, aux pompes injectant l'eau de fonctionner : on parle alors de "Loss Of Off Site Power" (LOOP, début de l'AG en 3 ou 4 heures).

Lors des accidents de refroidissement APRP, la puissance résiduelle des produits de fission n'est plus extraite par l'eau entraînant la fusion du cœur. Le cœur fondu forme un mélange complexe constitué de dioxyde d'uranium UO_2 et de l'ensemble des matériaux présents dans la cuve (acier de structures, éléments des barres de commandes, gaines en zirconium, etc.) . Ce matériau, appelé *corium*, a une température avoisinant les 2850 K. Au cours de sa progression, il interagit avec les structures métalliques environnantes (gaines des crayons combustibles, barres de commandes, cuve, etc.) pour se relocaliser vers le fond de cuve. Au fond de celle ci, si de l'eau est présente, le corium peut se refroidir pour former un lit de débris. Ce lit de débris, n'étant plus refroidi, va alors fusionner pour reformer un bain de corium, c'est ce qui a été observé lors de l'AG de Three Mile Island (voir figure 1.3). Des flux de chaleur importants provenant du corium vont alors être imposés sur la cuve pouvant mener à sa rupture et à la propagation du corium dans le puits de cuve. Le corium va interagir avec le radier en béton du réacteur, l'ablater et potentiellement s'échapper vers l'extérieur. Les différentes étapes de la propagation du corium sont décrites figure 1.4. Toutes les connaissances phénoménologiques de la propagation du corium sont issues des différents programmes expérimentaux mais aussi du retour d'expérience des différents AG. Celui de Three Miles Island a permis une meilleure compréhension des phénomènes mis en jeu dans la cuve par exemple.

Lors de la propagation du corium, trois risques majeurs mettent en péril la troisième barrière de confinement du réacteur nucléaire, dernière barrière avant l'environnement extérieur au réacteur : si celle-ci cède, un relâchement de déchets radioactifs dans l'environnement devient possible. Ces trois risques sont situés dans le réacteur dans la figure 1.5.

L'interaction corium/eau. Aussi appelée explosion de vapeur, cette interaction se produit lorsque le corium rentre en contact avec l'eau dans la cuve ou dans le fond de cuve, une explosion due à la vaporisation instantanée de l'eau causée par le fort écart de température entre les deux matériaux est alors possible et peut fortement endommager les structures environnantes. Dans la figure, ce risque est représenté par α.

Le risque hydrogène. Un fort dégagement d'hydrogène est possible lors de l'oxydation des gaines des crayons combustibles, des structures métalliques contenues dans la cuve, de la structure de la cuve ou enfin du béton de la cuve. Si cet hydrogène atteint une concentration trop importante, il peut s'enflammer et une explosion peut alors endommager l'enceinte du réacteur. Dans la figure, ce risque est représenté par γ.

L'interaction corium/béton. Lorsque le corium arrive dans le puits de cuve, il rentre en contact avec le béton composant ce dernier. Cette interaction est à l'origine de l'ablation du béton et pourrait conduire, dans certains cas, au percement du radier et à la libération de produits radioactifs dans l'environnement. Dans la figure, ce risque est représenté par ϵ.

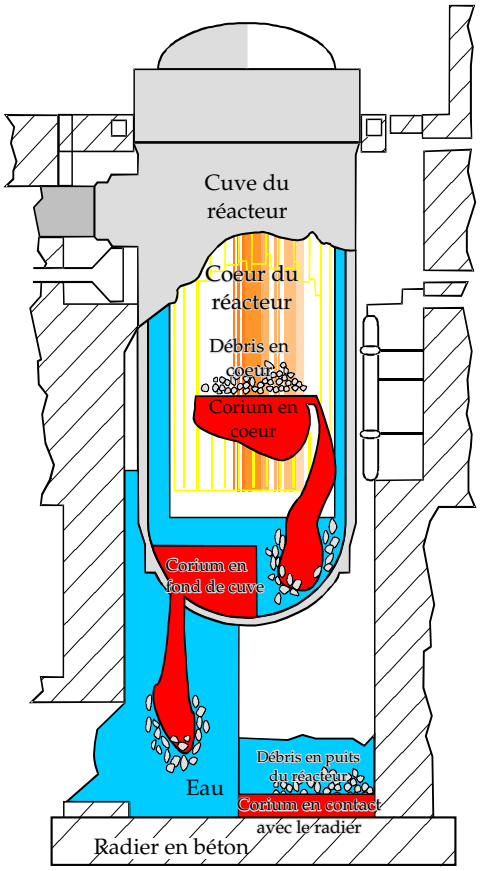


FIGURE 1.4 – Représentation schématique de la propagation du corium du cœur du réacteur jusqu'au radier en béton du puits du réacteur.

Pour prévenir l'apparition de ces phénomènes, une modélisation précise de la propagation du corium depuis la fonte du cœur jusqu'à son interaction avec le radier en béton du réacteur est *essentielle*. Ci-après sont présentés les différents types de modélisation utilisés dans le contexte des AG.

1.1.3 Les différents types de modélisation de la propagation du corium

Le développement de codes de calcul permettant de simuler le comportement du corium lors d'un AG est donc un enjeu majeur de la sûreté nucléaire. Ce développement a été accéléré par l'AG de Three Mile Island en 1979, d'abord aux États-Unis, puis en Europe. Toutefois, ce domaine est particulier en terme de modélisation et de simulation numérique :

Domaine multi-physique et multi-échelle. Des phénomènes *multi-physiques* et *multi-échelles couplés* interviennent lors de la propagation du corium dans le réacteur nucléaire. La modélisation de la propagation doit prendre

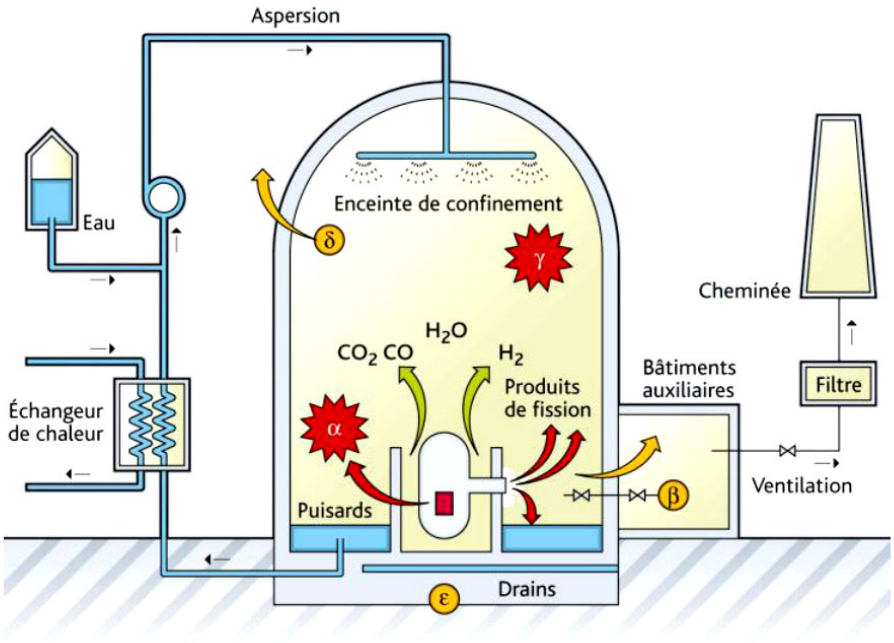


FIGURE 1.5 – Risques majeurs liés à la propagation du corium en et hors cuve. α représente l'explosion de vapeur, γ l'explosion d'hydrogène, ϵ l'interaction corium béton (extrait de http://www.irsn.fr).

en compte des phénomènes thermohydrauliques, thermomécaniques, thermochimiques, thermodynamiques, neutroniques etc. avec des échelles de temps allant de la seconde à l'année et des échelles de masse allant du gramme à la centaine de tonnes.

Différents besoins de simulation. Lors d'une étude de simulation de propagation, on peut vouloir se concentrer sur un phénomène physique précis, ou sur une partie d'un phénomène de propagation, ou sur la propagation du début de l'AG jusqu'à l'arrêt de l'AG dans tout le réacteur. Ainsi, suivant la granularité et le besoin de l'étude, la modélisation ne sera pas la même : en terme de temps et d'utilisation des ressources de calcul, la modélisation de tout l'AG ne pourra pas utiliser les mêmes modèles que la modélisation précise d'un phénomène physique.

Modélisation encore jeune et parfois compliquée. La connaissance physique de certains phénomènes est encore actuellement limitée rendant la modélisation compliquée. Il n'y a que peu de données réelles issues d'AG du fait du petit nombre de tels accidents et de la difficulté d'extraire des données de ceux-ci. De plus, les données extraites sont souvent post-mortem et aucune ou peu de données transitoires sont disponibles. Enfin, du fait des conditions extrêmes de température des AG ($T > 2500$ K),

les expériences sont difficiles à reproduire (mais réalisables). Lorsque des expériences sont possibles, elles ne sont pas nécessairement représentatives des échelles des réacteurs et sont faites avec des matériaux prototypiques ou simulants.

Besoin important de validation des codes. Les codes AG doivent être validés pour s'assurer qu'ils représentent de manière fidèle la réalité des phénomènes physiques et qu'ils permettent de simuler de manière fiable le déroulement des AG. La validation des codes de simulation et modélisation peut s'avérer compliquée. Des programmes expérimentaux ont été créés pour permettre d'avoir des données sur lesquelles valider les résultats de simulation des codes de calcul. La validation des codes se fait en trois étapes : en qualifiant séparément les modèles physiques implémentés dans le code, puis en qualifiant le code sur des essais intégraux, permettant de valider le couplage des différents modèles du code, puis en qualifiant le code sur les AG connus (l'AG de Three Miles Island par exemple).

Dans ce contexte particulier, on distinguera trois types de codes répondant chacun à une problématique bien précise :

Les codes dédiés. Ces codes ne servent à décrire qu'un ensemble restreint de phénomènes. Leur principal objectif étant d'approcher le mieux possible la physique pour en avoir une compréhension fine. Ces codes n'ont en général aucune restriction quant à la durée d'une simulation et peuvent donc se permettre d'être fortement maillés. On peut notamment citer des codes tels que `TONUS` (IRSN, France) [65], code CFD résolvant les équations de la thermohydraulique dans des géométries complexes 3D dans le cadre des risques liés aux explosions d'hydrogène, ou encore `MC3D` (IRSN, France) [76, 13] modélisant l'explosion de vapeur.

Les codes détaillés. Ces codes peuvent être utilisés pour simuler une partie ou l'ensemble d'un AG tout en restant proche de la physique avec des modèles précis. Ceux-ci peuvent donc avoir des temps de simulation long et sont surtout utilisés pour valider des codes moins précis ou pour dériver des modèles physiques plus simples à partir de modèles complexes. Par exemple, `ATHLET-CD` (GRS, Allemagne), `SCDAP/RELAP5` (INL, États-Unis) [88], ou enfin `ICARE/CATHARE` (IRSN, France) [47] résultant du couplage entre le code détaillé de simulation d'AG `ICARE2` et le code de thermohydraulique (1D et 3D) `CATHARE 2` sont de tels codes détaillés.

Les codes intégraux. Tels que `MAAP` (EPRI) [33, 34], `MELCOR` (Sandia National Laboratories, États-Unis) et `ASTEC` [17] (IRSN, France et GRS, Allemagne), ces codes permettent la simulation de l'ensemble de l'AG dans le réacteur. Ils sont en général composés de modules ou composants couplés entre eux représentant une certaine partie de la phénoménologie de l'AG ou un certain phénomène physique. Ceux-ci peuvent a

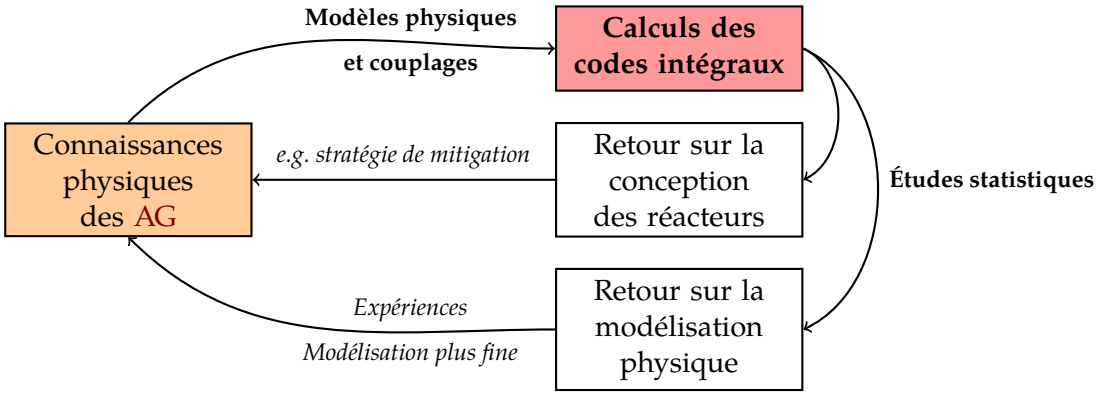


FIGURE 1.6 – Itération entre les connaissances physiques et phénoménologiques des AG et la simulation de ceux-ci par les codes intégraux couplée aux études statistiques.

priori prendre n'importe quelles formes. Les composants peuvent être d'autres codes, par exemple `ASTEC` utilise `ICARE` et `CESAR`, code de thermohydraulique, des modèles issus des résultats de codes fins ou des modèles simplifiés issus de corrélations.

Les codes dédiés et détaillés sont souvent utilisés en parallèle des codes intégraux. Ils peuvent servir de référence pour déterminer la validité des codes intégraux ou servir à étudier en profondeur un phénomène physique ou une partie de la phénoménologie de l'AG.

Du fait de la rapidité des calculs et du coût relativement faible des calculs, les codes intégraux rendent possibles les études statistiques d'incertitudes, ou "probability safety analysis" (PSA), par exemple des calculs avec des méthodes de Monte-Carlo. Ces études permettent d'identifier un ou plusieurs phénomènes physiques couplés ou un ou plusieurs paramètres physiques jouant un rôle important dans la phénoménologie de l'AG. Ceux-ci sont ensuite étudiés plus en profondeur par les codes détaillés, les codes dédiés ou des même des expériences si elles sont possibles.

Enfin, les codes intégraux permettent la simulation de l'*ensemble* de la phénoménologie de l'AG pour différents réacteurs et différents scénarios d'AG. Alliés aux études statistiques, les codes intégraux permettent un retour sur la bonne *conception des réacteurs* dans le cas de ces différents scénarios d'AG.

Ainsi, la combinaison des codes détaillés avec les codes intégraux permet d'améliorer les connaissances physiques et phénoménologiques des AG (voir figure 1.6). Cette évolution des connaissances phénoménologiques et physiques va de paire avec une forte évolution des modèles utilisés dans les codes intégraux. De ce fait, un code *scénario* AG statique et figé sur un réacteur et un scénario d'accident est inenvisageable. Dans le contexte des AG, un code

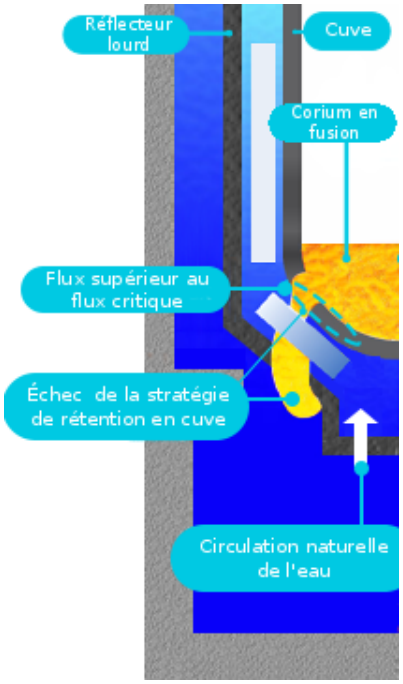


FIGURE 1.7 – Schéma de la stratégie de rétention en cuve du corium par renoyage du puits de cuve.

scénario doit offrir *modularité* et l'*évolutivité*.

La plate-forme logicielle PROCOR est un code intégral développé pour répondre aux besoins de modularité et d'évolutivité du contexte AG. C'est l'outil numérique utilisé pour cette thèse. Il est présenté ci-après.

1.2 Le code intégral PROCOR

1.2.1 Motivations de la plate-forme

L'une des raisons du développement de la plate-forme était de permettre la prise en compte des phénomènes transitoires en cuve [68]. Ces phénomènes sont d'importances dans le cadre de démonstration de réussite de la stratégie de mitigation des AG par la rétention du corium en cuve ou "in vessel retention" (IVR). La rétention du corium en en cuve est étudiée depuis les années 1980 avec l'AP600, l'AP1000 [26], et encore aujourd'hui dans le cadre du projet européen H2020/IVMR (2015-2019). Elle consiste à renoyer le puits de cuve du réacteur pour prévenir le percement de la cuve sous l'effet des flux de chaleur importants imposés par le corium en fond de cuve (voir figure 1.7). Pour des réacteurs de faibles puissances, approximativement 600 MWe, la démonstration de réussite de la stratégie IVR par des modèles *stationnaires* a été faite

par différents codes AG. Toutefois, pour des réacteurs de puissances plus élevées, supérieur à 1000 MWe, une démonstration plus robuste avec prise en compte des phénomènes transitoires est nécessaire [14].

L'existence de la plate-forme est aussi due à un mauvais retour d'expérience du code scénario LEONAR. Le code LEONAR [93] a été développé de 2007 à 2013 par le CEA. C'est un code intégral scénario initialement prévu pour les REP 1300MWe (réacteur de génération 2 ou Gen2) et basé sur l'analyse scénario REP1300. Il permet, à partir de données de dégradation du cœur issues du code MAAP, d'évaluer les risques de percement de la cuve du réacteur et de simuler la propagation du corium dans le puits de cuve. Dans les réacteurs de génération 3 (Gen3), une structure permettant le contrôle de la neutronique du cœur a été rajoutée : le réflecteur lourd. Dans le code LEONAR, un modèle correspondant à ce réflecteur lourd a été introduit. Néanmoins, l'introduction logicielle et l'utilisation du modèle de réflecteur lourd se sont avérées laborieuses du fait du manque de modularité et d'évolutivité de LEONAR. Ceci a permis de mettre en évidence les insuffisances du code : il est peu réutilisable pour de nouvelles analyses scénarios, difficile à maintenir et difficile à faire évoluer.

Issu de la volonté de corriger ce manque de modularité et d'évolutivité, et pour pallier le problème de modélisation physique dans le cadre de la stratégie IVR de mitigation des AG, le développement de la plate-forme logicielle PROCOR (PROpagation du CORium) commença en 2013.

1.2.2 Plate-forme de simulation et d'études statistiques

Le code PROCOR, code intégral, est associé à une démarche de recherche et de développement basée sur une modélisation simplifiée des phénomènes physiques avec prise en compte des incertitudes sur les modèles. C'est une plate-forme dans le sens où c'est une librairie d'outils, de fonctionnalités et de modèles. Elle est développée en Java et respecte le paradigme de programmation orientée objet dans le but de maximiser la généricité et l'évolutivité des fonctionnalités mises en place.

L'ensemble des objets de la plate-forme PROCOR permettent la construction d'*applications industrielles* dédiées à la propagation du corium lors d'un AG. L'ensemble des modèles physiques et leurs couplages constituant les applications sont issus d'analyses scénarios. Les différentes applications actuellement dans PROCOR correspondent à différents scénarios et différentes générations de réacteur (Génération 2 ou GEN2, Génération 3 ou GEN3).

La plate-forme est structurée en deux parties :
— Une partie *physique* déterministe : *c'est le cadre de la thèse*.
— Une partie *statistique* par l'association de variables aléatoires aux paramètres d'entrée d'une application physique déterministe. Les calculs

statistiques sont ensuite gérés par le code statistique CEA URANIE [43] couplé avec le logiciel ROOT. La partie statistique de la plate-forme n'est pas utilisée pour la thèse.

Pour faciliter les études statistiques, une attention particulière a été portée sur la gestion des paramètres d'entrée des modèles, et la gestion des équations, des états et des transitions entre les états des modèles, pour la réutilisation de modèles et équations déjà existants. Une description plus précise de l'architecture du code et des nouveautés apportées durant la thèse sera donnée par la suite (voir chapitre 3).

Dans la partie physique, la mutualisation des services, des modèles et de la démarche de développement d'applications permet de gagner en robustesse, en facilité et en rapidité pour le développement de nouvelles applications dédiées à de nouveaux réacteurs, notamment par la création d'une bibliothèques de modèles qui peuvent être versionnés. À partir d'une analyse scénario et phénoménologique, la création d'une application industrielle de simulation de propagation du corium lors d'un AG revient à :

— sélectionner un ensemble de modèles déjà existants ou en créer des nouveaux à partir des outils fournis par la plate-forme.
— définir le couplage entre les modèles.

Les modèles de la bibliothèque constituant les applications peuvent a priori prendre n'importe quelle forme. Par exemple, certains modèles sont décrits par des équations analytiques ou par des équations différentielles ordinaires correspondant à des équations de conservations de la masse, de l'énergie ou de l'enthalpie. D'autres peuvent prendre la forme de modèles quasi-stationnaires, modèles réduits (méta modèles ou réseaux de neurones) ou même faire appel à des logiciels externes.

1.2.3 Le couplage des modèles dans les applications industrielles

Les applications créées suivent un cycle de vie de développement figure 1.8 cohérent avec celui décrit précédemment figure 1.6. Celui-ci permet de capitaliser le savoir gagné au cours du cycle sur la phénoménologie et la modélisation physique des phénomènes importants. Durant ce cycle, chaque modèle physique de la plate-forme a un cycle de vie de développement spécifique : analyse physique et statistique, développement et implémentation logicielle, vérification unitaire (JUnit [58] en Java). Le cycle de vie et d'évolution d'une application PROCOR 1.8 justifie le fait que les modèles physiques la constituant doivent pouvoir être aisément modifiés ou changés par d'autres modèles.

Il est **inenvisageable** que l'application soit structurée en un seul bloc indivisible de modèles. La modification d'une telle application nécessiterait sa reconstruction complète, ce qu'il faut absolument éviter (risques d'erreurs lors de la reconstruction, croissance du temps de développement sur les évolutions, etc.). Une application doit "percevoir" les modèles physiques qu'elle

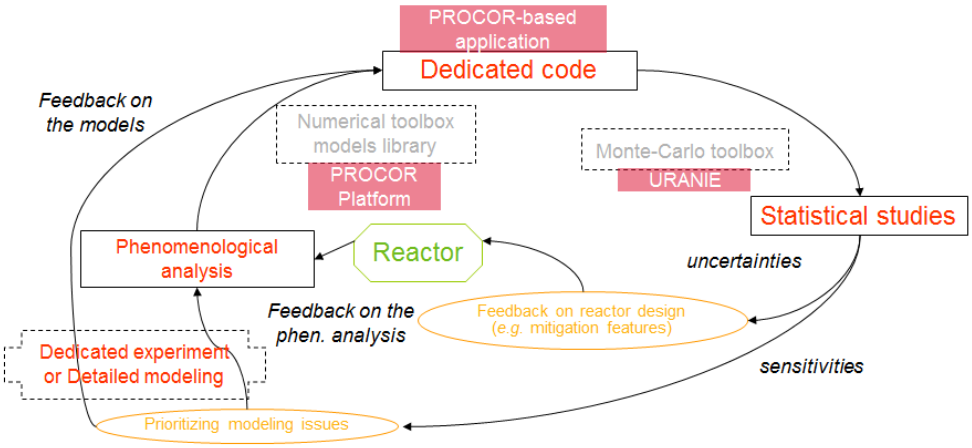


FIGURE 1.8 – Cycle de vie d'une application PROCOR. L'analyse statistique d'une application PROCOR, elle même issue de l'analyse scénario, permet d'avoir d'une part un retour sur la conception des réacteurs, et d'autre part un indicateur ou un retour sur les points de l'analyse scénario à améliorer et sur les parties de la modélisation physique à décrire plus finement.

utilise comme des *boîtes noires* avec des entrées et des sorties. Les applications peuvent ainsi aller piocher dans un large éventail de modèles et les coupler entre eux, tant que le couplage est compatible en terme d'entrées et de sorties. Ainsi, une application prend la forme d'un système complexe [11] composée d'un ensemble de modèles physiques couplés.

Le couplage entre les modèles d'une application débouche naturellement sur une résolution par un schéma en temps à deux niveaux : un pas de temps local à chaque modèle noté Δt et un pas de temps global noté $\Delta t \geq \delta t$. On parlera alors de macro boucle en temps pour la boucle en temps externe aux modèles.

1.2.4 Boucles en temps enchâssées et pathologies numériques

Néanmoins, le couplage par blocs de modèles dans les applications PROCOR introduit des pathologies numériques liées aux deux niveaux de boucle en temps. L'idéal serait d'avoir les avantages de modularité et d'évolutivité offert par ce type de couplage tout en effaçant la présence des deux niveaux de boucle en temps. La précision de calcul d'une application serait alors la même que la précision de résolution en interne des modèles. Toutefois, une résolution "naïve" du couplage ne permet pas d'effacer les deux niveaux de boucle en temps. Par résolution naïve, on entend par exemple un *chaînage semi-explicite* des modèles : ceux-ci sont résolus l'un après l'autre et ne communiquent qu'à chaque macro pas de temps. À l'intérieur de la macro boucle en

temps, les modèles ne peuvent avoir connaissance de l'état des modèles avec lesquels ils sont couplés. Une latence est ainsi introduite par le découplage et celle ci est naturellement dépendante de la longueur de la macro boucle en temps.

Généralement et sauf cas particuliers, par exemple le code ASTEC qui utilise un schéma de couplage prédicteur correcteur, c'est le chaînage semi-explicite qui est utilisé dans les codes intégraux pour les AG. Avant la thèse, c'était ce chaînage semi-explicite qui était utilisé dans la plate-forme PROCOR pour les applications industrielles de simulation de la propagation du corium lors d'AG. En raison de contraintes industrielles, le développement de la plate-forme PROCOR s'est concentré sur la création d'outils de base et de modèles physiques plutôt que sur la création d'une architecture entièrement dédiée au couplage et à la synchronisation de modèles.

Les pathologies numériques liés aux deux boucles en temps enchâssées sont présentées au travers d'exemples numériques issus de la plate-forme PROCOR d'*avant thèse*.

1.2.5 Les pathologies numériques sur des exemples

1.2.5.1 Sur des cas de tests industriels

Les résultats utilisés ci après sont issus d'une série de tests effectués par l'équipe PROCOR [6] qui permettent la comparaison du code PROCOR avec le code intégral MAAP4 [33]. L'objectif est de comparer le transitoire thermique et thermochimique d'un bain de corium en fond de cuve (voir figure 1.9) lors d'ajout sur sa surface supérieure d'acier provenant de la fusion de la cuve, de structures internes au réacteur ou de coulées métalliques ou oxydes provenant du cœur.

Le bain de corium peut être polyphasique suivant sa composition et composé d'une couche oxyde, métal lourd ou métal léger entourés d'une croûte. Au dessus des la croûte, une couche d'acier peut être présente. Cette configuration du bain stratifié en fond de cuve est décrite figure 1.9 (cette configuration est réutilisée dans le chapitre 4 et décrite dans la figure 4.19). Les différences de densité engendre une stratification du bain. Cette stratification évolue (il peut y avoir des inversions de stratification) et modifie les transferts d'énergie. Il en résulte l'ablation des structures dont la partie fondue intègre le bain ce qui modifie la structure du bain. De ce fait, un couplage existe entre la thermique et la thermochimie du bain.

Cette phénoménologie est importante pour la mitigation des AG dans le cadre de la stratégie IVR. En effet, elle peut mener à de nombreux cas de percement de la cuve par un phénomène de concentration de flux dit de Focusing Effect et engendrer la rupture de la cuve, l'une des trois barrières physiques de confinement du réacteur. Ce phénomène caractérise une dissymétrie entre

la puissance évacuée par transfert radiatif par la surface supérieure et par conduction par la surface latérale de la couche d'acier : lors du Focusing Effect, la majeure partie de la puissance est évacuée par la surface latérale car le transfert radiatif est beaucoup moins efficace, créant, lors d'une couche mince d'acier, une concentration du flux de chaleur sur cette surface. Ce phénomène est d'autant plus marqué lorsque la couche d'acier est fine.

Dans un fond de cuve de rayon R_{pool} de 2 m (voir figure 1.9), des coulées d'acier fondu viennent s'ajouter à différents instants par le dessus du bain stratifié. La configuration initiale correspond à un bain en équilibre thermo-

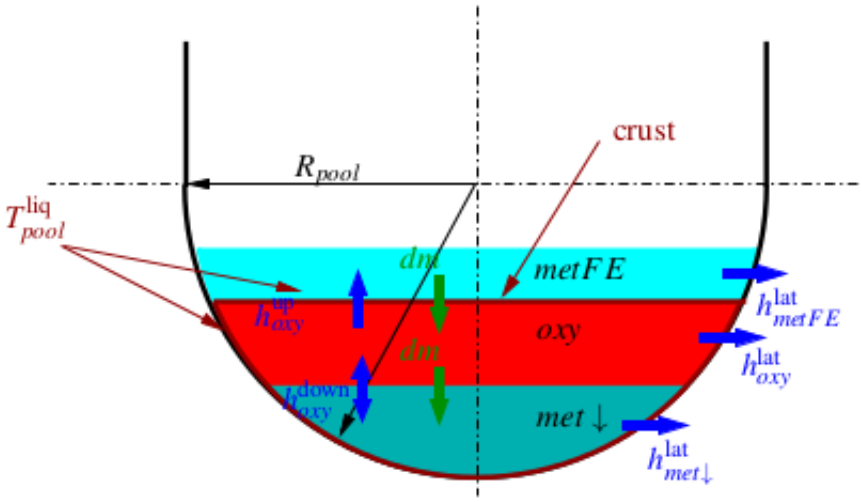


FIGURE 1.9 – Géométrie du bain stratifié en deux couches en équilibre thermochimique et thermique et surmonté d'une couche métallique hors équilibre séparée par une croûte (les flèches bleus représentent les transferts thermiques et les flèches vertes les sens des transferts massiques considérés par hypothèse)

chimique constitué d'une couche métallique lourde ($met\downarrow$) surmontée d'une couche oxyde (oxy) en régime permanent.

L'algorithme de résolution est séparée en quatre parties calculées l'une après l'autre à chaque macro pas de temps :

Calcul du modèle de thermochimie. Un équilibre thermochimique dépendant du débit d'acier que la couche d'oxyde reçoit est calculé et donne l'état d'équilibre à l'instant $t + \Delta t$ vers lequel le bain va converger.

Calcul du modèle de cinétique. Suivant l'état d'équilibre calculé, les transferts de masses inter-couches sont calculés.

Calcul du modèle de thermique. Le bilan thermique de chacune des couches

Δt (s)	1	10	50	100	200
m_{metFE} (kg)	8970	8962	8953	8777	10000
$t_{\text{inv.}}$ (s) en s	49634	49650	49700	49700	49000

TABLE 1.1 – Comparaison des masses finales de la couche d'acier hors équilibre et du temps de détection de l'inversion de la stratification.

est calculé permettant d'obtenir les flux de chaleur imposés par le bain sur les structures environnantes.

Calcul de l'ablation des structures. Les masses des structures ablatées sont calculées et ajoutées au bain de corium.

La simulation s'arrête à l'inversion de stratification dans le bain. On compare la masse m_{metFE} d'acier hors équilibre, couche supérieure du bain stratifié responsable du Focusing Effect, ainsi que le temps $t_{\text{inv.}}$ d'inversion de la stratification. On utilise pour simulation de référence le cas où le macro pas de temps Δt vaut 1 s. Le tableau 1.1 donne les résultats des calculs pour différents macro pas de temps $\Delta t \in \{10, 50, 100, 200\}$. On constate un écart sur la masse finale m_{metFE} allant jusqu'à plus de 1 tonne, soit près de 10%. Des conclusions peuvent être tirées :

— Même pour des macro pas de temps Δt bien inférieurs aux temps caractéristiques des différents modèles, $\approx$ 1800 s pour la thermochimie et $\approx$ 1000 s pour la thermique, le schéma de couplage n'a pas convergé. Il faut fortement réduire le macro pas de temps pour arriver à convergence. Ceci n'est pas envisageable dans le contexte présent pour des raisons évidentes de performance, par exemple inenvisageable pour des études statistiques avec un nombre important de calculs.

— Le découplage du modèle de thermochimie et du modèle de thermique implique une non cohérence de l'état des deux modèles à la fin du pas de temps Δt de calcul sur la macro boucle en temps. Le calcul de la thermochimie dépend du calcul de thermique, et le calcul de thermique dépend du modèle de thermochimie.

— Le temps d'inversion ne peut être connu qu'aux instants de la macro boucle en temps, une erreur sur son calcul de l'ordre de Δt est donc commise.

Pour effacer les deux niveaux de boucle en temps, diminuer l'impact du schéma de couplage et ne retrouver qu'au maximum l'erreur commise en interne des modèles, il est nécessaire d'avoir un schéma permettant :

— d'une part une communication dans les *deux sens* entre les modèles permettant un retour du modèle de thermique vers le modèle de thermochimie.

— d'autre part une synchronisation *à l'intérieur* de la macro boucle temps entre les modèles.

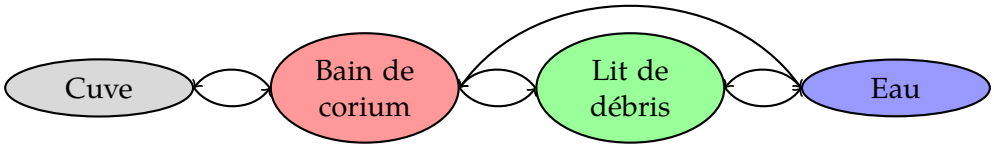


FIGURE 1.10 – Graphe de couplage partiel de l'application PROCOR TransientInVessel.

Δt (s)	1	2	10	50	100	200
m_{pool} (kg)	82252	82334	81154	80552	76599	70193
$t_{\text{fail.}}$ (s)	25436	25438	25430	25450	25300	25000

TABLE 1.2 – Comparaison des masses finales de corium et du temps de rupture de la cuve dans l'application TransientInVessel de la plate-forme

1.2.5.2 Sur une application industrielle PROCOR

L'application industrielle PROCOR TransientInVessel permet la simulation de la propagation du corium en cuve uniquement. La simulation démarre à l'instant de formation d'un bain de corium dans le cœur jusqu'au percement du fond de cuve ou la stabilisation des bains de corium en cœur et en fond de cuve. L'application est constituée entre autres d'un modèle de bain de corium, un modèle d'ablation de la cuve et du réflecteur lourd et un modèle de lit de débris et un modèle d'eau qui sont couplés. Les modèles de l'application et les couplages sont représentés par le graphe donné figure 1.10.

La simulation est divisée en plusieurs parties calculées l'une après l'autre à chaque macro pas de temps. Les deux parties principales sont :

1. Calcul du modèle de bain de corium : à partir de l'état du réacteur au début de macro pas de temps, des flux de chaleur sont calculés et imposés sur les structures environnantes du réacteur et les lits de débris.
2. Calcul des autres modèles : les structures peuvent être ablatées et les lits de débris chauffés peuvent fondre, des débits de masse sont calculés et ajoutés au bain de corium.

L'application a été calculée pour différents macro pas de temps. Les résultats sont donnés dans le tableau 1.2. On y compare les valeurs de masse de corium m_{pool} lorsqu'il y a rupture de la cuve, ainsi que le temps $t_{\text{fail.}}$ de cette rupture pour $\Delta t \in \{1, 2, 10, 50, 100, 200\}$. A la rupture de la cuve, on constate une dispersion de près de 10% de la masse de corium m_{pool} suivant la valeur du macro pas de temps utilisé. Cependant, cette masse conditionne la valeur du flux de chaleur imposé sur la cuve et la puissance résiduelle du bain de corium (puissance massique associée aux produits des fissions). Une

telle dispersion n'est pas acceptable pour une application industrielle potentiellement utilisée dans le cadre de la sûreté nucléaire.

Des conclusions similaires au précédent test peuvent encore être déduites :
— La convergence sur le macro pas de temps est effective pour des pas de temps petits en comparaison des temps caractéristiques des différents modèles qui sont de l'ordre de 1000 s. Réduire autant le macro pas de temps n'est pas envisageable pour une plate-forme industrielle prévue, entre autres, pour des études statistiques.
— Le découplage du modèle de bain de corium et des autres modèles peuvent amener l'ensemble du système dans un état non cohérent physiquement. Pendant un certain laps de temps, borné par le macro pas de temps, le bain de corium peut "voir" une structure ou un lit de débris qui peuvent avoir changé d'état, par exemple disparu.
— Le temps de rupture de la cuve ne peut être connu qu'aux instants de la macro boucle en temps, une erreur sur son calcul de l'ordre de Δt est donc commise.

Un schéma similaire à celui évoqué précédemment, i.e. permettant un retour entre les modèles et une synchronisation à l'intérieur de la macro boucle en temps, est nécessaire.

De ces deux exemples et du retour d'expérience sur d'autres applications industrielles, il est absolument nécessaire de munir la plate-forme PROCOR d'une *architecture logicielle dédiée au couplage* permettant d'utiliser des *schémas numériques de couplage et de synchronisation* des modèles plus précis. Ce sont les objectifs de la thèse. Ils sont décrits ci-après dans la section 1.3.

1.3 Objectifs de la thèse

La thèse est divisée en trois étapes (les objectifs) : *étudier, adapter ou créer des schémas de couplage et de synchronisation de modèles* (à la section 1.3.1), puis *créer une architecture logicielle de couplage et y implémenter les schémas de couplage et de synchronisation* (à la section 1.3.2), enfin *tester l'architecture sur des problèmes couplés AG identifiés dans la plate-forme PROCOR* (à la section section 1.3.3). Ces trois objectifs sont détaillés ci-après.

1.3.1 Schéma de couplage et synchronisation

Bien que répondant aux besoins exprimés par le contexte des accidents graves en matière de modularité et d'évolutivité des modèles d'une application, le couplage des modèles sur deux niveaux de boucle en temps introduit des pathologies numériques et des erreurs qui entachent, voir invalides, les résultats. Ces pathologies doivent être traitées et résolues par des schémas de couplage et de synchronisation plus complexes qu'un *simple* chaînage semi-explicite des modèles. Issus directement de la plate-forme, les deux exemples

précédent le montrent (voir la section 1.2). Deux pathologies essentielles ressortent des calculs précédents :

Les pathologies liées au couplage. Les modèles couplés doivent pouvoir communiquer dans les deux sens permettant ainsi un retour d'informations lors de la résolution sur un macro pas de temps (ce que le chaînage semi-explicite ne permet pas).

Les pathologies liées à la synchronisation. Les modèles couplés doivent pouvoir être capables d'informer les autres modèles de leur changement d'état ou de leur disparition, et donc de se synchroniser à l'intérieur de la macro boucle en temps qui doit s'adapter automatiquement en fonction du ou des événements apparaissant à l'intérieur de la macro boucle en temps.

De la littérature sont extraits un ensemble de schémas de couplage. Ces schémas sont *adaptés* pour répondre aux besoins spécifiques de la plate-forme en matière de couplage et de synchronisation. Ceci est présenté dans le chapitre 2.

1.3.2 Architecture logicielle et algorithmes de résolution

Les schémas de couplage et de synchronisation doivent être implémentés dans la plate-forme PROCOR. Pour des raisons et contraintes industrielles de temps, le développement de la plate-forme ne s'est pas concentré sur la résolution numérique du couplage. Aucune architecture dévouée au couplage de modèles n'est prévue dans la plate-forme originelle (d'avant thèse). Les modèles sont couplés en "dur" dans chaque application et ce couplage est fixe. De fait, l'implémentation des schémas s'est divisé en deux étapes :

La création d'une architecture logicielle destinée au couplage de modèles. Entre autres, elle définit de nouvelles classes de modèles qui supportent des opérations nécessaires au bon fonctionnement des schémas de couplage et de synchronisation, par exemple la sauvegarde et la restauration. Elle est pensée pour s'intégrer progressivement, de manière transparente dans la plate-forme et en respectant les contraintes de *non-régression* imposées sur la plate-forme industrielle.

L'implémentation des algorithmes de résolution du couplage sous cette architecture. Ceux-ci correspondent aux schémas de couplage et de synchronisation proposés dans la section 2.2. Ils sont "traduits" informatiquement et intégrés dans la nouvelle plate-forme de couplage et de synchronisation en utilisant les fonctionnalités offertes par la nouvelle architecture.

De nombreuses plate-formes de couplage existent déjà. Par exemple dans le domaine du nucléaire, la plate-forme ALLIANCES [80] développée par la CEA, ANDRA et EDF pour la simulation des risques liés au traitement et au recyclage des déchets nucléaires ou dans le domaine de l'interaction fluide-structure la plate-forme preCICE [42]. Néanmoins, le contexte AG fait que l'environnement de couplage qu'il faut développé est particulier :

— la modélisation incertaine des AG fait que les modèles constituant le couplage sont amenés à *évoluer* et doivent être fréquemment *changés* au sein du couplage.
— le schéma de couplage *dépend* du scénario de l'AG et de la phénoménologie associée qui sont souvent incertains.
— les modèles couplés peuvent être *variés* et très *différents*, par exemple des modèles 0D, 1D, 2D, etc. (voir l'annexe A pour des exemples de modèles 0D).
— les modèles peuvent apparaître ou disparaître au cours de la simulation rendant le graphe de couplage des modèles *dynamique*, par exemple le percement de la cuve du réacteur ou l'évaporation de l'eau dans le réacteur amenant les modèles associés à disparaître.
— les modèles peuvent avoir plusieurs états et aux instants des transitions d'états, des discontinuités apparaissent et il faut absolument un *système de synchronisation* sur ces discontinuités.

La description de l'architecture logicielle créée ainsi que les algorithmes implémentés dans la plate-forme au cours de la thèse sont présentés au chapitre 3.

1.3.3 Les problèmes couplés de la plate-forme PROCOR

Des problèmes clés, récurrents et problématiques de couplage et de synchronisation de la plate-forme sont identifiés pour l'analyse numérique et les calculs de cette thèse. Ils sont représentatifs de la grande diversité en matière de modélisation qu'il est possible de trouver dans la plate-forme PROCOR dans le contexte des AG. Ils permettent de mettre en évidence les pathologies numériques citées précédemment et de mettre à l'épreuve les nouveaux schémas de la plate-forme.

Les problèmes identifiés de la plate-forme PROCOR sont constitués de plusieurs modèles couplés. À chaque modèle est associé un solveur. Ces solveurs sont ensuite couplés sur la macro boucle en temps par le schéma de couplage et synchronisation. Différents types de solveurs existent (un solveur 0D ou point (voir l'annexe A pour des exemples de modèles 0D), un solveur maillé, un solveur stationnaire ou quasi-stationnaire, un solveur à base de méta modèles ou de réseaux de neurones, un solveur appelant un code externe, un solveur hybride mélangeant 0D,1D, 2D et 3D, etc.). Ils sont associés à des couplages et des événements différents.

Par conséquent, du fait de la grande variété de solveurs, on essayé de catégorisé les différents types de couplages et d'événements.

Les deux premières catégories correspondent à des particularités de couplage. Le couplage entre les modèles sera dit :

hétérogène en espace s'il couple des phénomènes physiques modélisés et discrétisés avec différents niveaux de raffinement en espace. Par exemple, un problème de conduction de la chaleur entre deux domaines avec un solveur maillé 1D pour un domaine et un solveur avec une modélisation point, ou 0D ou "lumped parameter".

hétérogène en temps s'il couple des phénomènes physiques modélisés avec différents niveaux de raffinement en temps. Par raffinement en temps, il est entendu des modèles transitoires, des modèles quasi-stationnaires ou des modèles stationnaires. Par exemple, un problème de conduction de la chaleur entre deux domaines avec un solveur considérant que l'état stationnaire est atteint sur le domaine et un solveur transitoire.

Cette catégorisation des couplages vient de la littérature. Les couplages hétérogènes en espace sont des problèmes récurrents, notamment étudiés dans [37]. Le couplage hétérogène en temps est important dans le contexte AG, comme expliqué dans [68, 14].

Les deux dernières catégories correspondent à des particularités de synchronisation. Un événement que doit détecter le schéma de synchronisation sera dit :

continu si la transition de l'état avant l'événement à l'état après l'événement se fait de manière continue : les solveurs et fonctions mathématiques sous-jacentes passent d'un état à l'autre sans discontinuité. C'est le cas par exemple pour un événement mineur pour lequel les équations des deux états sont les mêmes : la transition d'un état à l'autre se fait continûment.

discontinu si la transition n'est pas continue. C'est le cas par exemple pour un événement majeur pour lequel les équations des deux états sont complètement différentes auquel cas des discontinuités liées à l'initialisation du nouvel état à partir de l'ancien état peuvent apparaître (voir [82] ou [74] par exemple).

De même, ces catégories sont issues de la littérature. La détection et le traitement de ces événements sont importants et sont encore un sujet de recherche, comme expliqué dans [74].

On se concentre sur trois problèmes couplés : *la conduction de la chaleur dans plusieurs domaines couplés* (décrit section 4.1), *la vidange d'une couche par plusieurs débits de masse couplés* (décrit section 4.2) et *le couplage des modèles d'une application industrielle* `PROCOR` (décrit section 4.3). Ces trois problèmes permettent de parcourir l'ensemble de toutes les catégories décrites précé-

	Couplage hétérogène en espace	Couplage hétérogène en temps	Synchro. sur événement continu	Synchro. sur événement discontinu
Problème de conduction	oui	oui	oui	non
Problème de la vidange	non	oui	non	oui
Problème d'une application PROCOR	oui	oui	oui	oui

TABLE 1.3 – Catégorisation des problèmes couplés identifiés de la plate-forme PROCOR.

demment. Le tableau 1.3 résume et catégorise les particularités en matière de couplage et synchronisation des problèmes couplés identifiés de la plate-forme PROCOR.

Sur les trois problèmes couplés, des résultats numériques ainsi que des analyses numériques (lorsqu'elles sont possibles) sont donnés dans le chapitre 4 d'analyses et de calculs numériques.

Chapitre 2

Vue d'ensemble des schémas de résolution du couplage et de synchronisation des modèles

Chaque phénomène physique intervenant lors de la propagation du corium dans un réacteur nucléaire lors d'un AG est modélisé par un modèle $\mathcal{M}_i$. Ce phénomène peut être purement géométrique, par exemple le déplacement d'une interface, ou un purement physique, par exemple le calcul d'une variable thermodynamique.

À chaque modèle est associé un solveur numérique. Par la suite, $\mathcal{M}_i$ désignera à la fois le modèle ou son solveur associé. Chaque modèle a sa propre approche de modélisation. Par exemple une modélisation fine maillée, une modélisation plus grossière, une modélisation à base de métamodèles, etc. (implémentée directement dans la plate-forme PROCOR ou faisant appel à un code externe à la plate-forme). Le contexte AG, expliqué section 1.2, impose une modélisation pas blocs couplés des phénomènes physiques. Formellement, l'ensemble des modèles modélisant la propagation prend la forme d'un système complexe (voir [11] pour plus de détails sur les systèmes complexes) constitué de plusieurs solveurs en interaction. Il est représenté schématiquement à la figure 2.1.

Dans ce chapitre sont données d'une part un état de l'art et d'autre part les avancées apportées par la thèse pour :

— la résolution du couplage entre les modèles $\mathcal{M}_i$ section 2.1.
— la synchronisation entre les modèles $\mathcal{M}_i$ section 2.2.

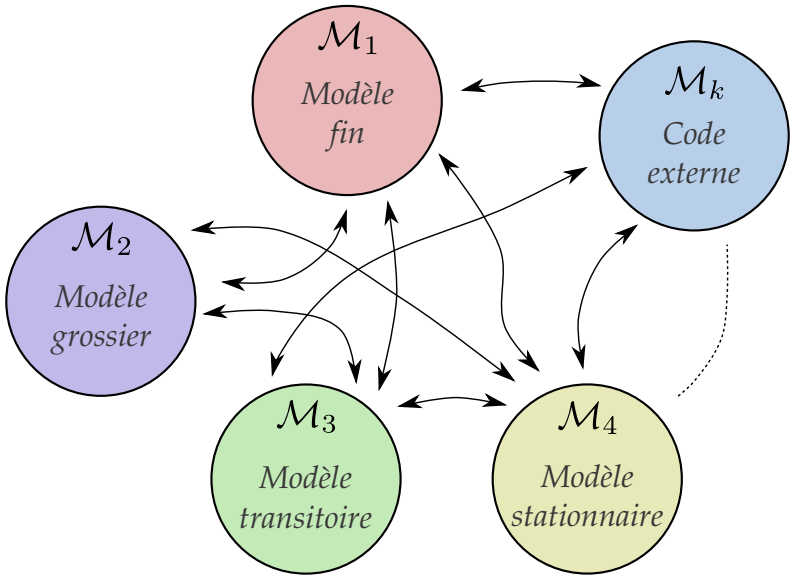


FIGURE 2.1 – Système complexe constitué de plusieurs solveurs modélisant les phénomènes intervenant lors de la propagation du corium lors d'un AG. Les flèches représentent des interactions (des couplages) entre les solveurs.

2.1 L'approche partitionnée pour la résolution du couplage des modèles

L'approche *partitionnée* [35] imposée par le contexte AG est beaucoup utilisée dans d'autres domaines et pour d'autres applications. De fait, un formalisme et un vocabulaire existent déjà. On peut aussi la trouver sous le nom de *"staggered approach"* dans [27, 30, 29], *"loose coupling"* dans [32] ou *"segragated approach"* dans [51].

Dans cette approche, les solveurs $\mathcal{M}_i$ sont vus comme des entités en interaction, mobiles et interchangeables du système complexe résolus dans un certain ordre.

Une autre méthode de résolution des modèles couplés est l'approche *monolithique*, que l'on peut aussi trouver sous le nom *fully-coupled approach* dans [50]. Dans cette approche, les équations des modèles $\mathcal{M}_i$ sont rassemblées dans un système d'équations global et toutes les équations sont résolues par le même schéma.

Le découpage du problème initial en blocs de modèles dans l'approche partitionnée rappelle les méthodes de *"splitting"*. Il est toutefois suggéré dans [36] que ces méthodes s'avèrent moins générales et applicables à un partitionnement de plus bas niveau d'un modèle, par exemple suivant certaines directions de l'espace.

La résolution de problèmes couplés peut aussi être faite par des méthodes de *"Waveform Relaxation"*. Ces méthodes sont des méthodes itératives de résolution de problèmes couplés consistant à envoyer, à chaque macro pas de temps, des données temporelles entre les modèles couplés. Elles sont particulièrement adaptées lorsque les modèles sont fortement sous-cyclés. De nombreux résultats théoriques sont disponibles, notamment sur la convergence super linéaire des itérations. Plus de détails peuvent être trouvés dans [55], [7], [18] ou [52].

Une revue des techniques de résolution de couplage de modèles peut être trouvée dans [62].

Dans cette section, on se concentre sur l'*approche partitionnée*.

2.1.1 Quelques mots sur l'approche monolithique

Concrètement, pour une résolution monolithique des modèles couplés, un solveur global $\mathcal{M}$ et une variable global $\mathbf{u}$ est construit par concaténation des solveurs $\mathcal{M}_i$ et de leurs variables $\mathbf{u}_i$. Ils sont donnés par les équations suivantes :

$$\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_k)^t \tag{2.1}$$

$$\mathcal{M} = (\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_k)^t \tag{2.2}$$

La résolution des modèles couplés revient alors à la résolution de l'équation :

$$\mathcal{M}(\mathbf{u}) = \mathbf{0}. \tag{2.3}$$

Si $\mathcal{M}$ est linéaire, l'équation (2.3) devient

$$\begin{pmatrix} \mathcal{M}_1 & \mathcal{M}_{12} & \dots & \mathcal{M}_{1k} \\ \mathcal{M}_{21} & \mathcal{M}_2 & \dots & \mathcal{M}_{2k} \\ & & \vdots & \\ \mathcal{M}_{k1} & \mathcal{M}_{k2} & \dots & \mathcal{M}_k \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_k \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_k \end{pmatrix} \tag{2.4}$$

dans laquelle apparaissent les matrices hors diagonales représentant les couplages entre les différents solveurs. Le système à résoudre est linéaire et des méthodes de résolutions des systèmes linéaires classiques sont utilisées. Toutefois, l'équation (2.3) n'est généralement pas linéaire et des méthodes de Newton sont utilisées (voir [50] par exemple).

En pratique, la reconstruction du solveur global $\mathcal{M}$ à partir des solveurs $\mathcal{M}_{ij}$ est difficile, voire impossible. La construction du solveur $\mathcal{M}$ soit se faire dans l'autre sens chronologique : le solveur $\mathcal{M}$ est construit puis on y intègre des solveurs $\mathcal{M}_{ij}$ développés spécialement pour cette occasion. La réutilisation de modèle déjà existant ou l'appel d'un code externe devient compliqué.

Dans l'ensemble, la résolution des modèles couplés par l'approche monolithique est peu maintenable et peu évolutive. Elle est incompatible avec les contraintes imposées par le contexte AG.

2.1.2 L'approche partitionnée en comparaison à l'approche monolithique

La résolution partitionnée a été étudié par C.A. Felippa et K.C. Park dans [35] ou dans une version plus récente [36]. Dans ces papiers, les auteurs explicitent clairement les avantages et inconvénients de cette approche par rapport à l'approche monolithique, et identifient les points clés permettant de choisir entre l'une ou l'autre des deux approches. Adaptée dans le contexte AG pour la modularité et l'évolutivité qu'elle offre, l'approche partitionnée présente d'autres intérêts par rapport à l'approche monolithique. Elle permet notamment :

Une modélisation indépendante des modèles particulièrement intéressante lors du travail de développement et modélisation au sein d'une équipe développement (par exemple l'équipe PROCOR). Le travail en groupe ne doit être fait qu'en amont du développement pour spécifier les entrées et sorties des modèles pour assurer la compatibilité du couplage entre les modèles. En tant qu'entités mobiles et non statiques dans le système complexe 2.1, les modèles peuvent être vus comme des boîtes noires avec des entrées et sorties. Tant qu'il y a compatibilité entre ces entrées et sorties, les modèles peuvent être échangés avec d'autres modèles compatibles et donc *développés indépendamment*.

La réutilisation de code et de modèle existant. Par exemple pour les applications industrielles dans PROCOR, le modèle de bain de corium sera utilisé pour la modélisation de la propagation en cuve, hors cuve et dans le réacteur (voir la figure 1.4). De plus, la réutilisation de modèles ne se limite pas nécessairement à des modèles de la plate-forme PROCOR. L'appel à un logiciel externe peut être caché à l'intérieur d'un solveur de la plate-forme et le même code externe peut être appelé plusieurs fois au travers de la plate-forme.

Un contrôle de la granularité du partitionnement du problème monolithique. La granularité peut être choisie librement en découpant plus ou moins finement le problème initial. Les blocs ainsi découpés seront couplés lors de la résolution par la résolution partitionnée. Chaque bloc devient un solveur $\mathcal{M}_i$ du système complexe. Il peut alors subir un traitement spécifique en fonction de ses contraintes et besoins. Chaque bloc peut ainsi utiliser un schéma de discrétisation spatiale et temporelle adapté à la dynamique du problème physique sous-jacent. Cette notion est particulièrement intéressante dans le contexte AG avec des modèles multi-physiques et multi-échelles : des entités allant de quelques grammes

à plusieurs tonnes et des temps caractéristiques de l'ordre de 10^{-3} s pour la neutronique dans le corium jusqu'à 10^3 s pour les phénomènes conducto-convectifs devront être considérées. À terme, on espère que les gains en précision et rapidité apportés par ce traitement spécifique seront plus importants que les pertes en précision causées par le couplage des modèles. Le schéma de couplage doit permettre d'effacer au maximum le couplage des modèles pour au final récupérer au maximum l'erreur commise en interne des solveurs $\mathcal{M}_i$.

Dans [36], C.A. Felippa et K.C. Park soulignent la complexité que peut cacher le partitionnement du problème (c'est à dire son découpage en blocs). Bien qu'attrayante, la définition du partitionnement est un point crucial de l'approche partitionnée et elle peut s'avérer compliquée. La création du système complexe donné à la figure 2.1 et des solveurs $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k$ à partir du problème initial en un seul bloc se fait par :

Un partitionnement correspondant à une décomposition en espace, par exemple fluide et structure ou une décomposition physique, par exemple thermique et thermochimique, du système. Dans le cas de l'interaction fluide-structure, le partitionnement correspondant à la décomposition du modèle monolithique regroupant le fluide et la structure en deux modèles, l'un pour le fluide, l'autre pour la structure, qui seront des **partitions** du problème partitionné. Une fois ce découpage effectué, comme représenté dans la figure 2.2, on obtient un ensemble de modèles, ou partitions, dont la réunion est le modèle initial (en un seul bloc). Ce partitionnement peut être récursif (les partitions sont elles mêmes partitionnées) créant ainsi un arbre de partitions.

Un découpage en temps ou "splitting" correspondant à une décomposition en temps des partitions permettant de définir des sous-cycles temporels d'une partition à l'autre. La notion de sous-cycle temporel est particulièrement intéressante dans le cas de problème multi-échelle avec des problèmes physiques aux dynamiques parfois très différentes.

Du fait du partitionnement, certaines propriétés des solveurs, ou partitions, deviennent externes ou partagées (*"interfield properties"*). D'autres propriétés des solveurs restent internes ou privés (*"intrafield properties"*). Ces propriétés peuvent prendre la forme de quantités scalaires ou vectorielles : ce sont les variables de couplage du système complexe (par exemple, lors d'un couplage fluide structure, les solveurs partagent à l'interface une force et une pression s'appliquant à l'interface). Cette terminologie est étroitement liée à celle du domaine des systèmes répartis et calculs parallèles pour lesquels la notion de ressources partagées et d'accès à ces ressources est primordiale.

Du fait de ce partage de ressources, deux niveaux de parallélisme sont possibles :

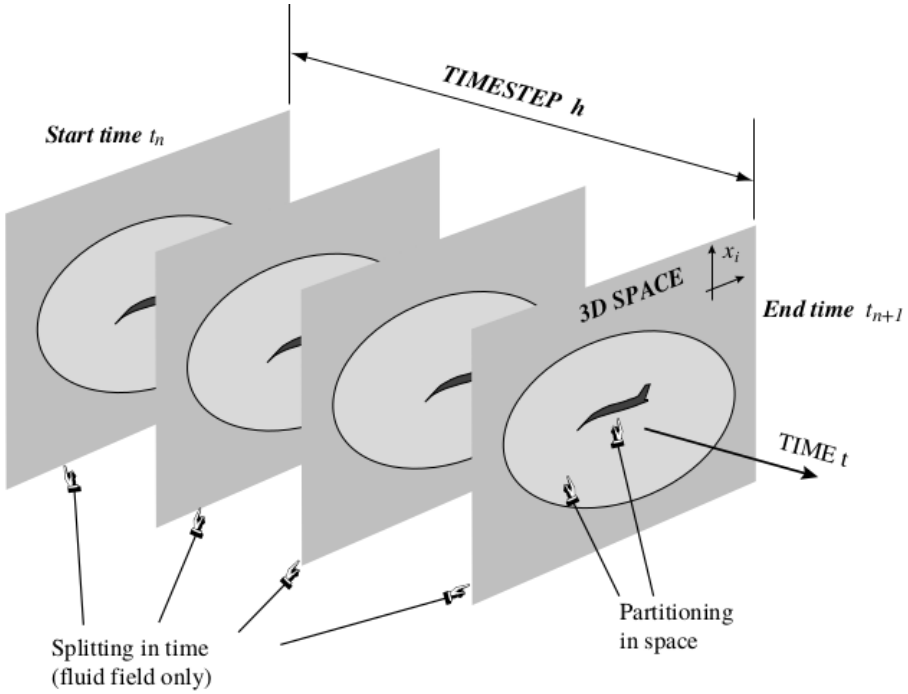


FIGURE 2.2 – Partitionnement et découpage en temps du problème initial en différents modèles couplés. Extrait de [36]

Un parallélisme externe géré par le schéma de couplage permettant d'avancer en temps en parallèle l'ensemble des partitions en gérant correctement le partage des ressources et l'échange des variables de couplage.

Un parallélisme interne entièrement géré par le solveur en interne et inaccessible depuis l'extérieur (par exemple par le schéma de couplage).

Cependant, les auteurs soulignent dans [36] que le parallélisme externe peut parfois être difficile à mettre en œuvre.

Cependant, l'approche partitionnée présente certains problèmes et risques dont il faut être conscient :

— Pour des interactions fortes entre modèles, les *gains en temps* et ressources par rapport à la résolution monolithique ne sont pas assurés. Cependant, lorsque les interactions entre modèles sont faibles, la résolution partitionnée est généralement plus intéressante (bien que l'approche monolithique peut être adaptée, voir [51] par exemple). Le choix de l'approche à adopter doit se faire au cas par cas.

— L'approche monolithique permet généralement de gérer avec plus d'aisance les instabilités de couplage entre modèles et est réputée plus *robuste* (voir [21, 51] par exemple). Néanmoins, les avancées récentes sur les approches partitionnées (voir [21] par exemple) permettent d'ap-

pliquer cette approche à des couplages qui ne pouvaient être traités avant que par des approches monolithiques.
— L'utilisation de différents solveurs, bien qu'attrayante car ceux-ci peuvent être échangés aisément, peut amener dans une situation d'*incompatibilité* entre solveurs. Une étape de post-traitement, ou "data-mapping" est alors nécessaire (voir [19] par exemple), par exemple une projection entre des maillages différents. Cette étape de post-traitement peut être coûteuse et avoir des conséquences sur la stabilité numérique des calculs et/ou sur la conservation de certaines quantités physiques.

2.1.3 Formalisme mathématique de couplage

Dans l'approche partitionnée proposée dans [36], des schémas de résolution de problèmes de couplage comme celui de la figure 2.1 sont proposés. Néanmoins, pour introduire de tels schémas, un formalisme mathématique doit être introduit. Les notations introduites permettront d'une part de définir formellement le problème de couplage et d'autre part de manipuler des objets couplés, les modèles (ou partitions, ou solveurs), qui peuvent prendre a priori des formes très variées, en particulier dans le contexte des AG.

2.1.3.1 Le graphe de couplage

De manière générale, tout système complexe issu d'un problème de couplage de modèles peut être représenté par un graphe de couplage. Par exemple, le problème de couplage figure 2.1 est représenté par le graphe de la figure 2.3 (en ne considérant que les couplages d'un modèle $\mathcal{M}_i$ parmi les k modèles). Les nœuds du graphe correspondent aux modèles $\mathcal{M}_j$ du problème, pour $j \in [1, k]$. Chaque solveur manipule différents types de données :

les données internes au solveur : son vecteur d'état $\mathbf{u}_i$ pour $i \in [1, k]$, on dira qu'elles appartiennent au domaine Ω_i associé à $\mathcal{M}_i$.

les données partagées entre solveurs : les vecteurs $\mathbf{b}_{ij}$ pour $i, j \in [1, k]$, on dira qu'elles appartiennent à l'interface Γ_{ij} entre $\mathcal{M}_i$ et $\mathcal{M}_j$. Chaque modèle $\mathcal{M}_i$ reçoit et envoie des données de et vers des solveurs : ce sont les variables de couplage appelées "interfield properties" dans [36]. On distinguera les variables de couplage d'entrées et de sorties. Celles calculées et envoyées par $\mathcal{M}_j$ vers $\mathcal{M}_i$ sont des variables d'entrées de $\mathcal{M}_i$, notées $\mathbf{b}_{ji}$. Celles calculées et envoyées par $\mathcal{M}_i$ vers d'autres modèles $\mathcal{M}_j$ sont des variables de sorties, notées $\mathbf{b}_{ij}$. Les communications entre solveurs sont représentées sur le graphe par un arc orienté du modèle calculant vers le modèle recevant.

les conditions limites du problème : les vecteurs $\mathbf{b}_i$ pour $i \in [1, k]$. Elles appartiennent à la partie de la frontière de Ω_i partagée avec aucun autre

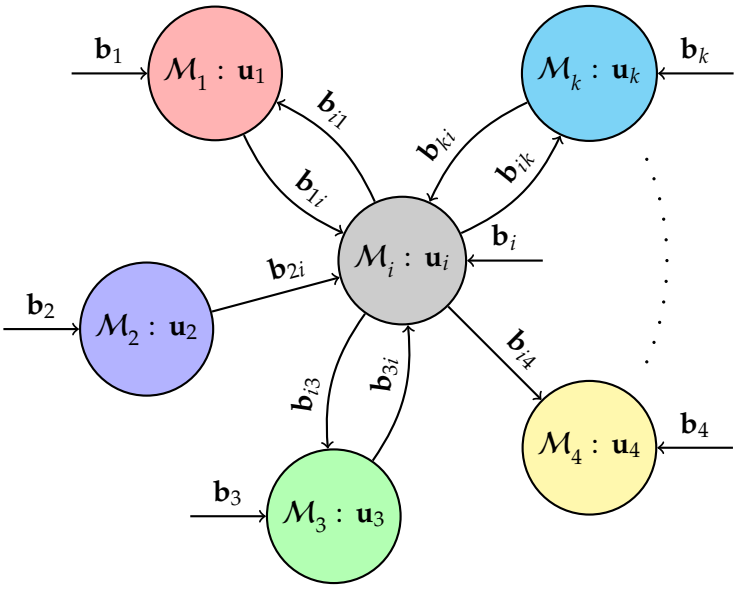


FIGURE 2.3 – Exemple de graphe de couplage partiel d'un système complexe associé à un problème de couplage. Le système est composé de k modèles. Le graphe est centré sur le modèle $\mathcal{M}_i$ et seuls ses couplages sont représentés.

solveur du problème mais seulement avec la frontière $\partial\Omega$ du problème lui même. Elles appartiennent donc à $\partial\Omega_i \cap \partial\Omega$.

Les notations relatives à la localité des données, provenant de la décomposition de domaine [24], sont résumés figure 2.4. Le terme *"localité"* employé est purement symbolique : les domaines et interfaces peuvent ne pas correspondre à un découpage géométrique du domaine. Le découpage peut être fait en termes de phénomène physique auquel cas l'interface n'est pas associée à une partie géométrique du domaine mais à une variable physique (par exemple une température ou une pression partagées et calculées par plusieurs solveurs).

Le solveur $\mathcal{M}_i$ est couplé en entrée avec un ensemble de solveurs $\mathcal{M}_j \in N_{\star i}$ ($n_{\star i} = |N_{\star i}|$) avec

$$j \in N_{\star i} \overset{\text{def.}}{=} \{l \ / \ \exists\, \mathbf{b}_{li}\}.$$

Il est couplé en sortie avec un ensemble de solveurs $\mathcal{M}_j \in N_{i\star}$ ($n_{i\star} = |N_{i\star}|$) avec

$$j \in N_{i\star} \overset{\text{def.}}{=} \{l \ / \ \exists\, \mathbf{b}_{il}\}.$$

On notera l'ensemble des variables de couplage d'entrées reçues par le modèle $\mathcal{M}_i$

$$\{\mathbf{b}_{ji}\}_{j \in N_{\star i}} \overset{\text{def.}}{=} (\mathbf{b}_{j_1 i}, \ldots, \mathbf{b}_{j_{n_{\star i}} i}) \overset{\text{def.}}{=} \mathbf{b}_{\star i}$$

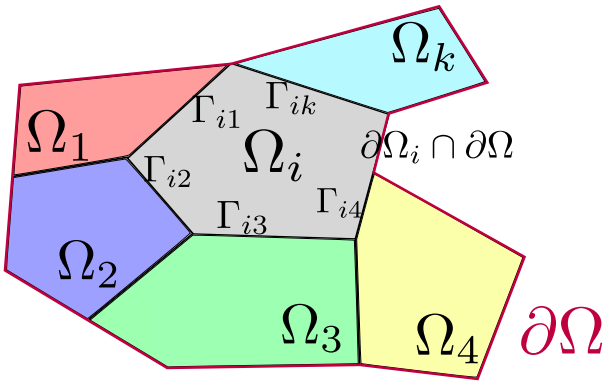


FIGURE 2.4 – Localité des données manipulées par le solveur $\mathcal{M}_i$ du problème de couplage : les données peuvent être internes au modèle, c.a.d $\in \Omega_i$, partagées entre solveurs, c.a.d. $\in \Gamma_{ij}$, ou des conditions limites du problème, c.a.d. $\in \partial\Omega_i \cap \partial\Omega$.

et l'ensemble des variables de couplage calculées et envoyées par le modèle $\mathcal{M}_i$

$$\{\mathbf{b}_{ij}\}_{j\in N_{i\star}} \overset{\text{def.}}{=} (\mathbf{b}_{ij_1}, \ldots, \mathbf{b}_{ij_{n_{i\star}}}) \overset{\text{def.}}{=} \mathbf{b}_{i\star}.$$

2.1.3.2 Les équations discrètes couplées

Les solveurs $\mathcal{M}_i$ constituant le problème de couplage de la figure 2.3 sont discrétisés en espace et en temps en interne et couplés en temps à un niveau supérieur externe aux modèles (la macro boucle en temps Δt). Ceci donnant les deux niveaux de boucle en temps :

— Les modèles gèrent eux-mêmes leur schéma interne de discrétisation spatiale et/ou temporelle. Par exemple, le schéma de discrétisation temporelle peut être explicite, implicite et utiliser diverses méthodes (schéma d'Euler, un schéma de Runge-Kutta d'ordre élevé ou des schémas multi-pas). Le schéma est supposé adapté au problème physique résolu par le solveur. On notera δt le micro pas de temps utilisé par les solveurs pour le schéma de discrétisation temporelle.

— Les modèles sont résolus entre les temps $t^0, t^1, \ldots, t^n, \ldots$ de la macro boucle en temps et communiquent à chacun de ces temps. On notera $\Delta t > \delta t$ le macro pas de temps de la macro boucle en temps. Ce macro pas de temps peut varier au cours de la simulation.

On représente les équations discrétisées du solveur $\mathcal{M}_i$ par une fonctionnelle $\mathcal{F}_i$. Elle prend en paramètre le vecteur d'état $\mathbf{u}_i$ du modèle $\mathcal{M}_i$, les variables de couplage reçues $\mathbf{b}_{\star i}$ par ce solveur ainsi que les conditions limites $\mathbf{b}_i$ données par le problème. Avec ces notations, la résolution sur un macro

pas de temps du couplage des modèles est donnée par les équations

$$\begin{cases} \mathcal{F}_1(t^n,\ \Delta t,\ \mathbf{u_1},\ \{\mathbf{b}_{j1}(\mathbf{u}_j,\ \mathbf{b}_{\star j})\}_{j\in N_{\star 1}},\ \mathbf{b}_1) = 0 \\ \mathcal{F}_2(t^n,\ \Delta t,\ \mathbf{u_2},\ \{\mathbf{b}_{j2}(\mathbf{u}_j,\ \mathbf{b}_{\star j})\}_{j\in N_{\star 2}},\ \mathbf{b}_2) = 0 \\ \qquad\qquad \vdots \\ \mathcal{F}_i(t^n,\ \Delta t,\ \mathbf{u_i},\ \{\mathbf{b}_{ji}(\mathbf{u}_j,\ \mathbf{b}_{\star j})\}_{j\in N_{\star i}},\ \mathbf{b}_i) = 0 \\ \qquad\qquad \vdots \\ \mathcal{F}_k(t^n,\ \Delta t,\ \mathbf{u_k},\ \{\mathbf{b}_{jk}(\mathbf{u}_j,\ \mathbf{b}_{\star j})\}_{j\in N_{\star k}},\ \mathbf{b}_k) = 0. \end{cases} \tag{2.5}$$

Les interactions entre modèles apparaissent dans (2.5) par le biais des variables d'entrées $\mathbf{b}_{ji}(\mathbf{u}_j,\ \mathbf{b}_{\star j})$ à l'interface Γ_{ji} entre les modèles $\mathcal{M}_i$ et $\mathcal{M}_j$ pour $j \in N_{\star i}$. Ces variables d'interfaces sont calculées lors de la résolution du modèle $\mathcal{M}_j$. Pour être calculées, elles ont besoin de son état $\mathbf{u}_j$ et de ses variables d'interfaces d'entrées $\mathbf{b}_{\star j}$. A leurs tours, ces variables d'interfaces peuvent être calculées par d'autres modèles nécessitant des données du modèle $\mathcal{M}_i$ ou même directement par le modèle $\mathcal{M}_i$.

Dans le système d'équations (2.5), le macro pas de temps de couplage Δt apparaît comme une inconnue du problème. En effet, celui-ci peut être calculé par une des fonctionnelles $\mathcal{F}_i$ si un événement apparaît lors de la résolution forçant le solveur $\mathcal{M}_i$ à stopper sa résolution avant la fin complète du macro pas de temps. Le fait que Δt soit une inconnue du problème couplé permettra notamment la *synchronisation des modèles* (présentée ci-après dans la section 2.2). Par la suite, la macro pas de temps Δt n'apparaît plus dans les équations pour simplifier les notations, mais il est *toujours une inconnue du problème*.

2.1.3.3 Les équations de point fixe et les résidus aux interfaces

De fait, dans (2.5), les dépendances entre modèles n'apparaissent que par les variables de couplage d'entrées et de sorties. Les modèles, ou solveurs, peuvent être assimilés à des boîtes noires qui sont des entités closes et fermées recevant des valeurs d'entrées et calculant des valeurs de sorties.

Mathématiquement, le solveur $\mathcal{M}_i$ peut être vu comme une fonction calculant les variables de sorties $\mathbf{b}_{i\star}$ à partir des variables d'entrées $\mathbf{b}_{\star i}$. Dans sa définition, seules les variables d'interfaces apparaissent. Le solveur $\mathcal{M}_i$ est définit par l'équation

$$\mathbf{b}_{i\star} = \mathcal{M}_i(\mathbf{b}_{\star i}) \tag{2.6}$$

ou de manière équivalente

$$\{\mathbf{b}_{il}\}_{l\in N_{i\star}} = \mathcal{M}_i(\ \{\mathbf{b}_{ji}\}_{j\in N_{\star i}}\ ). \tag{2.7}$$

Grâce à (2.7), le problème de couplage entre le modèle $\mathcal{M}_i$ et le modèle $\mathcal{M}_j$ peut être défini seulement à l'aide des variables transitant par les interfaces

Γ_{ij} et Γ_{ji} entre les modèles. À cette interface, les équations de couplage sont données par

$$\mathbf{b}_{ij} = \mathcal{P}_{ij} \circ \mathcal{M}_i(\mathbf{b}_{ji}, \{\mathbf{b}_{li}\}_{l\in N_{\star i},\, l\neq j}) \tag{2.8}$$
$$\mathbf{b}_{ji} = \mathcal{P}_{ji} \circ \mathcal{M}_j(\mathbf{b}_{ij}, \{\mathbf{b}_{lj}\}_{l\in N_{\star j},\, l\neq i}), \tag{2.9}$$

avec $\mathcal{P}_{ij}$ et $\mathcal{P}_{ji}$ les projecteurs aux interfaces Γ_{ij} et Γ_{ji} définis par

$$\mathcal{P}_{ij}(\mathbf{b}_{i\star}) \overset{\text{def.}}{=} \mathbf{b}_{ij}$$
$$\mathcal{P}_{ji}(\mathbf{b}_{j\star}) \overset{\text{def.}}{=} \mathbf{b}_{ji}.$$

Les équations (2.8) et (2.9), centrées sur les interfaces entre $\mathcal{M}_i$ et $\mathcal{M}_j$, mettent en lumière le principe d'action-réaction à l'interface entre ces deux modèles. À l'image d'un ressort que l'on tire et relâche, une légère modification de la variable d'interface $\mathbf{b}_{ji}$ (représentant par exemple la position de l'interface) va impacter la variable d'interface associée $\mathbf{b}_{ij}$ (représentant par exemple la pression à l'interface) impactant l'interface qui se "déplace" dans un sens ou dans l'autre.

La force avec laquelle celle ci se déplace est dépendante de la force du couplage entre les modèles $\mathcal{M}_i$ et $\mathcal{M}_j$. Cette force peut être mesurer à l'aide des matrices Jacobiennes $\frac{\partial \mathcal{M}_i}{\partial \mathbf{b}_{ji}}$ et $\frac{\partial \mathcal{M}_j}{\partial \mathbf{b}_{ij}}$. L'évaluation de cette force peut être difficile car, suivant la forme et la nature du solveur, l'accès à ces matrices n'est pas tout le temps assuré.

En réunissant (2.8) et (2.9), le problème de couplage aux interfaces Γ_{ij} et Γ_{ji} entre $\mathcal{M}_i$ et $\mathcal{M}_j$ revient à la résolution d'une équation de point fixe donnée par

$$\mathbf{b}_{ij} = \mathcal{P}_{ij} \circ \mathcal{M}_i \left(\mathcal{P}_{ji} \circ \mathcal{M}_j \left(\mathbf{b}_{ij}, \{\mathbf{b}_{lj}\}_{l\in N_{\star j},\, l\neq i}\right), \{\mathbf{b}_{li}\}_{l\in N_{\star i},\, l\neq j}\right). \tag{2.10}$$

Cette formulation point-fixe est abondamment utilisée dans la littérature, par exemple dans [75]. Naturellement, la formulation est symétrique et peut être formulée à l'aide de la variable d'interface $\mathbf{b}_{ji}$.

À partir de l'équation de point fixe (2.10), on peut définir l'opérateur de résidu à l'interface Γ_{ij} entre $\mathcal{M}_i$ et $\mathcal{M}_j$ par

$$\mathcal{R}_{ij}(\mathbf{b}_{ij}) \overset{def}{=} \mathcal{P}_{ij} \circ \mathcal{M}_i \left(\mathcal{P}_{ji} \circ \mathcal{M}_j \left(\mathbf{b}_{ij}, \{\mathbf{b}_{lj}\}_{l\in N_{\star j},\, l\neq i}\right), \{\mathbf{b}_{li}\}_{l\in N_{\star i},\, l\neq j}\right) - \mathbf{b}_{ij} \tag{2.11}$$

pour une certaine valeur de $\mathbf{b}_{ij}$.

Le résidu $\mathcal{R}_{ij}(\mathbf{b}_{ij})$ exprime le déséquilibre qu'il peut exister à l'interface. Deux cas peuvent alors se présenter :

— Les conditions d'interfaces sont respectées et, dans ce cas, le résidu à l'interface est nul. On dira alors que les modèles $\mathcal{M}_i$ et $\mathcal{M}_j$ sont **fortement couplés**.
— Les conditions d'interfaces ne sont pas respectées et, dans ce cas, un déséquilibre existe et le résidu à l'interface n'est pas nul. On dira alors les modèles $\mathcal{M}_i$ et $\mathcal{M}_j$ sont **faiblement couplés**.

Ces termes sont issus de la littérature (voir [57] par exemple) et peuvent être utilisés aussi pour désigner la force physique du couplage entre les deux modèles. Il est important de distinguer un couplage fort numérique, lorsque le résidu donné par l'équation (2.11) est nul, d'un couplage fort physique lorsque l'interaction physique entre les deux modèles est importante. Il en va de même pour le couplage faible. On a donc deux types de couplage : un couplage physique et un couplage numérique.

Le couplage physique est dépendant des modèles et est imposé. Il est hors cadre de la thèse. Le couplage numérique, qui introduit une erreur s'il n'est pas résolu correctement, est, entre autres, le sujet de cette thèse.

2.1.4 Objectif du schéma de résolution du couplage

Dans l'approche partitionnée, la résolution du couplage des modèles permet d'obtenir une solution approximant celle obtenue par l'approche monolithique. Mathématiquement, cela signifie que la solution du problème de couplage obtenue par l'approche partitionnée en résolvant le système d'équations (2.5) approxime le mieux possible la solution obtenue par l'approche monolithique en résolvant l'équation (2.3). La résolution du couplage se fait par le parcours du graphe de couplage de la figure 2.3. Les nœuds du graphe (les modèles) sont visités et résolus dans un certain ordre, donné par le schéma de couplage, et communiquent les nouvelles variables de couplage calculées. Un schéma de couplage est donc la donnée d'un ordre dans lequel les solveurs du problème vont être appelés et comment ils vont communiquer leurs données.

Il est important de définir ce qu'est un bon schéma de couplage. Bien qu'étant fortement dépendant du problème considéré, une début de définition est possible. Tout au long de la revue bibliographique faite pour cette thèse, les schémas de couplage essaient généralement de trouver le meilleur compromis entre la *stabilité numérique* de résolution du couplage, étroitement liée à la précision des résultats donnés, et la *performance* en matière de temps de calcul et de ressources utilisées lors de la résolution du couplage. Un bon schéma de couplage résout le problème couplé de manière *stable* et *performante* et permet d'obtenir une solution *proche de celle obtenue par la résolution du couplage par l'approche monolithique*.

Remarques sur la stabilité numérique des schémas de couplage. Bien qu'un calcul stable ne donne pas nécessairement des résultats précis, un calcul instable donne nécessairement des résultats non exploitables. Dans [16, 45], il est clairement stipulé que la stabilité du schéma est dépendante de la force physique de l'interaction entre les modèles. Le schéma de couplage doit aussi respecté les contraintes en termes de macro pas de temps pour assurer la stabilité numérique des solveurs. Ces contraintes sont généralement imposées par le temps caractéristique de la physique sous-jacente à chaque modèle. Lorsqu'un modèle impose un macro pas de temps trop faible (une dynamique physique très faible par exemple), il devient bloquant dans la résolution couplage car il ralentit la résolution. Lorsqu'un modèle impose un macro pas de temps trop important (car il utilise une modélisation stationnaire par exemple), il peut rendre les résolutions d'autres modèles instables.

Remarques sur la performance des schémas de couplage. La taille du graphe de couplage ainsi que le nombre d'appels des solveurs par macro pas de temps vont grandement jouer sur les performance du schéma. Si un modèle doit être utilisé avec un macro pas de temps faible pour des raisons de stabilité, des sous-cycles peuvent être utilisées, comme dans [84]. Le modèle est résolu sur des plus petits pas de temps tandis que les autres restent sur le macro pas de temps initial. Le sous-cyclage permet de découplé les contraintes de stabilité imposées par les solveurs. De plus, pour améliorer les performances du schéma de couplage, l'approche partitionnée permet la parallélisation des calculs (voir la section 2.1.2).

Ces deux remarques illustrent bien la difficulté de juger de la qualité d'un schéma de couplage : un schéma pourra être jugé bon et adapté à un problème et pour autant mauvais et inutilisable pour un autre problème. Le choix du schéma est fortement dépendant du problème. Cependant, il est possible de classer les schémas de résolution du couplage en deux catégories :

— **Les schémas de couplage explicites** (SCE) [35, 28, 46, 31, 30, 29, 84, 83, 91, 9] qui ne nécessitent qu'un nombre déterminé d'appel des solveurs par macro pas de temps. Cependant, de tels schémas ne permettent qu'un couplage faible entre les modèles. L'équilibre imposé par de tels schémas est atteint avec une erreur de l'ordre du macro pas de temps de couplage [84]. De fait, certains auteurs parlent aussi de schéma permettant d'obtenir un couplage fort jusqu'à un certaine erreur proportionnelle au macro pas de temps.

— **Les schémas de couplage implicites** (SCI) [60, 57, 77, 44, 21, 61, 94, 22, 78, 41, 66] qui exécutent un processus itératif à chaque macro pas de temps appelant les solveurs jusqu'à ce qu'un certain critère de convergence soit satisfait. Si le processus a convergé, les modèles sont fortement couplée et la solution donnée par la résolution monolithique est retrouvée.

Le choix du schéma, c'est à dire explicite ou implicite, à utiliser est fortement dépendant de la force des interactions entre les modèles du problème. Lorsque l'interaction physique est trop importante, un SCI est conseillé [16]. Par contre, lorsque l'interaction est faible, un SCI a un impact trop important sur les performances du calcul pour donner des résultats sensiblement équivalent à ceux obtenus avec un SCE en matière de précision par rapport à la solution monolithique. Toutefois, cette interaction entre modèles peut être difficile à évaluer :

— Elle peut être évaluée **a priori** en ayant connaissance de la modélisation physique. Toutefois, celle ci peut changer et évoluer nécessitant d'adapter le schéma de couplage. C'est en particulier le cas dans le contexte des AG dans lequel la modélisation est en perpétuelle évolution.
— Elle peut être évaluée **a posteriori** à l'aide des matrices Jacobiennes du résidu aux interfaces donné par l'équation (2.11). Toutefois, ces matrices peuvent parfois être inaccessibles.

Conclusion sur la difficulté du choix du schéma de couplage. Dans un contexte de couplage de solveurs vus comme des boîtes noires, avec potentiellement des informations a priori et/ou a posteriori limitées à leur sujet, le choix du schéma de couplage peut s'avérer laborieux. Toutefois, il existe des moyens permettant de reconstruite des informations manquantes à partir du peu de données transmises par les solveurs. Par exemple, la reconstruction des matrices jacobiennes au fur et à mesure des itérations des SCI, introduites par la suite.

2.1.5 Représentation graphique du schéma de couplage

Un schéma de couplage correspond au parcours du graphe de couplage 2.3 :

— pour chaque nœud qu'il traverse, le schéma résout le modèle correspondant.
— une fois qu'il a parcouru ce nœud, le schéma demande au modèle de communiquer ses données aux modèles avec lesquels il est couplé, i.e. correspondant à des arcs du graphe de couplage.

Le schéma peut parcourir plusieurs fois le même nœud sur un même macro pas de temps, auquel cas le schéma sera implicite. Si tous les nœuds ne sont parcourus qu'une fois, le schéma sera explicite.

Pour chaque graphe de couplage, on peut avoir une représentation graphique du schéma de couplage sous la forme d'un *graphe de séquencement des modèles*. Considérons par exemple une partie (figure 2.5) du graphe précédent (figure 2.3) représentant le couplage entre les modèles $\mathcal{M}_1$, $\mathcal{M}_3$ et $\mathcal{M}_i$.

Pour ce graphe de couplage, la figure 2.6 donne des exemples de graphes de séquencement possibles entre les modèles $\mathcal{M}_1$, $\mathcal{M}_3$ et $\mathcal{M}_i$. De tels graphes

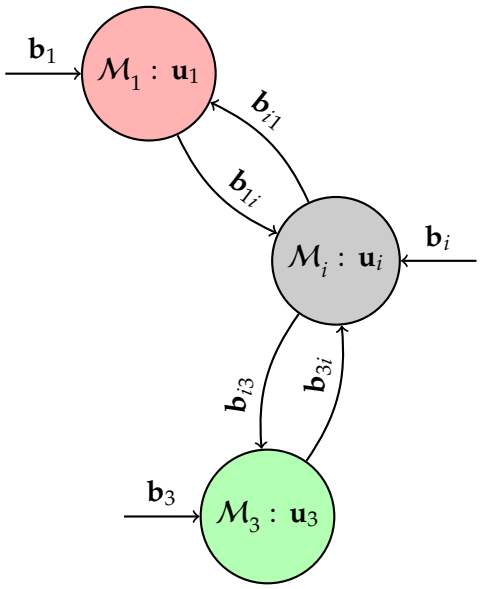


FIGURE 2.5 – Exemple de graphe de couplage entre les modèles $\mathcal{M}_1$, $\mathcal{M}_3$ et $\mathcal{M}_i$.

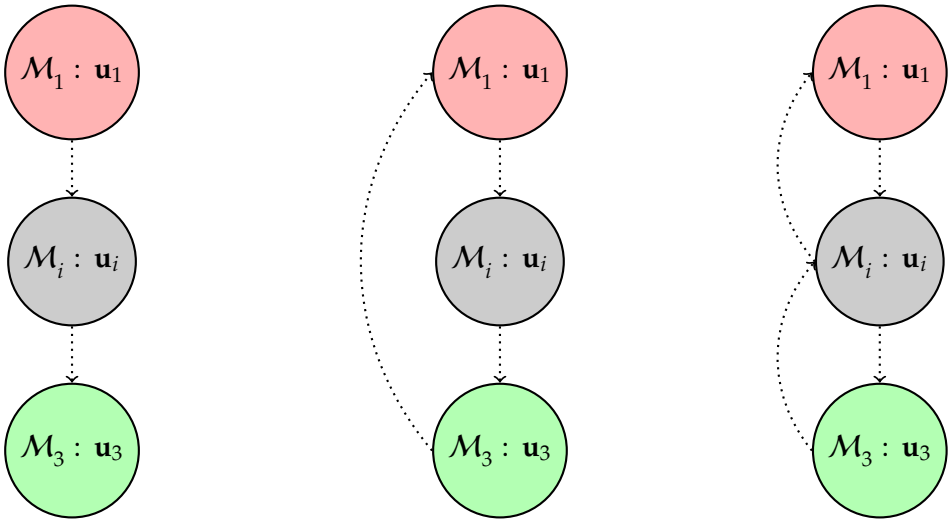


FIGURE 2.6 – Graphes de séquençage possibles pour la résolution sur un macro pas de temps du couplage entre les modèles $\mathcal{M}_1$, $\mathcal{M}_3$ et $\mathcal{M}_i$ donné par le graphe de couplage 2.5

représentent le séquençage *sur un macro pas de temps* entre les modèles pour résoudre explicitement (*à gauche dans la figure*) et implicitement (*au milieu et à droite dans la figure*) le couplage entre les modèles.

Le graphe de séquençage se lit de haut en bas. La résolution des modèles est faite du haut du graphe jusqu'au bas du graphe. La séquence est symbolisée par une flèche en pointillés qui *ne correspond pas forcément à un couplage* entre deux modèles mais seulement à un ordonnancement logique des modèles dans le schéma de résolution. Pour un SCE, chaque modèle est une seule fois sur un macro pas de temps. Pour un SCI, les modèles peuvent être résolus plusieurs fois sur un même macro pas de temps. Dans ce cas, des boucles apparaissent dans le graphe, représentées par une flèches partant du bas vers le haut. Dés que le SCI rentre dans une boucle, il itère jusqu'à convergence du couplage dans la boucle. Il s'arrête lorsque toutes les boucles ont convergé.

Comme illustré dans le graphe de séquencement donné par la figure 2.6, pour un même graphe de couplage, plusieurs graphes de séquencement sont possibles. Dans la section 4.3, partant d'un graphe de couplage plus complexe issu d'une application industrielle de la plate-forme `PROCOR`, plusieurs graphes de séquencement possibles sont donnés. Pour cette application, différents calculs et tests ont permis de faire un choix sur un séquencement de modèles satisfaisant en termes de précision et de performance. Néanmoins, ce choix est fortement dépendant du problème considéré.

L'obtention du séquencement optimal pour un graphe de couplage donné, i.e. celui trouvant l'équilibre optimal entre la précision et la rapidité de résolution du couplage, est un axe de recherche à explorer suite à cette thèse. Il est évoqué dans les ouvertures données à la fin de ce manuscrit.

2.1.6 Schémas de couplage explicite

Les schémas de couplage explicites (SCE) donnent une approximation de la solution du problème monolithique (2.3) en résolvant le problème partitionné (2.5) en ne faisant qu'un nombre déterminé et borné d'appels à chaque solveur par macro pas de temps. Cependant, de tels schémas ne permettent qu'un couplage faible entre les modèles. L'équilibre imposé par les SCE est atteint avec une erreur de l'ordre du macro pas de temps de couplage [84].

Les SCE restent néanmoins intéressant au regard de leur performance et de leur simplicité de mise en œuvre. Pour ces raisons, ils sont largement utilisés pour des applications diverses et variées. Ils ont été étudié et amélioré dans le cadre de l'interaction fluide structure dans [28, 46, 31, 30, 29]. Dans [84], S.Piperno étudie notamment la précision, la mise en parallèle et le sous-cyclage, et trouve des conditions de stabilité des SCE sur des cas simplifiés (géométrie 1D, équations linéarisées) des équations aéro-élastiques avant de tester ces conditions sur des cas plus complexes (géométrie 2D et 3D). Le SCE

est aussi utilisé pour résoudre des problèmes de couplage thermique structure dans [5] ou pour des problèmes de couplage thermo-hydro mécanique dans [87].

Dans [35, 36, 84, 83] sont introduits un ensemble de SCE : le schéma "Conventionnal Serial Staggered" (CSS), le schéma "Conventionnal Parallel Staggered" (CPS), le schéma "Improved Serial Staggered" (ISS), le schéma "Improved Parallel Staggered" (IPS) et le schéma "Generalized Serial Staggered" (GSS). "Staggered" signifie que la résolution des modèles est échelonnée ou décalée en temps. Le terme "time-lag" est aussi beaucoup utilisé dans ce contexte, signifiant que certains modèles sont résolus *après* d'autres modèles : un retard en temps se créé entres les modèles couplés.

"Conventionnal Serial Staggered" (CSS). Le schéma CSS correspond a une résolution semi-explicite des équations discrétisées (2.5), ou de manière équivalente des équations aux interfaces (2.7), du problème partitionné. Lors de la résolution sur un macro pas de temps par le schéma CSS, les solveurs ont à leur disposition des variables d'entrées évaluées à t^{n+1} ou à t^n suivant si le solveur calculant ses variables a déjà été résolu ou non. Avec ses données, les solveurs sont résolus et envoient leurs données toutes évaluées à t^{n+1} aux autres solveurs. On suppose par exemple que les solveurs $\mathcal{M}_i$ du problème figure 2.3 pour $i \in [1, k]$ sont appelés par le schéma dans l'ordre naturel. L'algorithme 1 donne les étapes de la résolution du problème de couplage par le schéma CSS de l'instant initial de simulation t^0 à l'instant final t^{n_f}.

Algorithme 1 Algorithme de résolution du schéma "Conventionnal Serial Staggered" (CSS).

for $n = 0$ to $n_f - 1$ **do**
  for $i = 0$ to k **do**
    $\{\mathbf{b}_{il}^{n+1}\}_{l \in N_{i\star}} = \mathcal{M}_i(\{\mathbf{b}_{ji}^{n+1}\}_{j<i,\, j \in N_{\star i}}, \{\mathbf{b}_{ji}^{n}\}_{j>i,\, j \in N_{\star i}})$./* Résolution de $\mathcal{M}_i$ */
    for $l \in N_{i\star}$ **do**
      envoyer $\mathbf{b}_{il}^{n+1}$ à $\mathcal{M}_l$.
    end for
  end for
end for

"Conventionnal Parallel Staggered" (CPS). Le schéma CSS peut être adapté naturellement pour utiliser une résolution totalement explicite des équations discrétisées (2.5) du problème partitionné (ou de manière équivalente, des équations aux interfaces (2.7)). Ce faisant, une résolution parallèle des solveurs devient possible. L'algorithme 2 donne les étapes de la résolution du

problème de couplage par le schéma CPS de l'instant initial de simulation t^0 à l'instant final t^{n_f}.

Algorithme 2 Algorithme de résolution du schéma "Conventionnal Parallel Staggered" (CPS).

```
1: for n = 0 to n_f − 1 do
2:     for i = 0 to k do /* boucle parallèle */
3:         {b_il^{n+1}}_{l∈N_{i⋆}} = M_i( {b_ji^n}_{j∈N_{⋆i}} ). /* Résolution de M_i */
4:     end for
5:     for i = 0 to k do /* boucle parallèle */
6:         for l ∈ N_{i⋆} do
7:             envoyer b_il^{n+1} à M_l.
8:         end for
9:     end for
10: end for
```

Appliqué au problème de couplage des équations fluide structure d'un piston dans [84], il est montré que le schéma CSS est toujours d'ordre un en temps. L'erreur de couplage commise est proportionnelle au macro pas de temps de couplage Δt et ceci peu importe la précision du solveur fluide et du solveur structure qui sont utilisés. En particulier, il est montré que :

- le schéma CSS peut être inconditionnellement stable dans la cas de solveurs implicites pour le fluide et la structure mais toujours d'ordre un en temps.
- si le solveur fluide est explicite et que le solveur structure est implicite, la stabilité du schéma CSS dépend de la stabilité du schéma du fluide. Le schéma CSS reste d'ordre un en temps.
- le sous-cyclage doit être traité avec beaucoup d'attention. Il est notamment montré que dans le cas de solveurs implicites pour le fluide et la structure, une stratégie naïve de sous-cyclage peut rendre le schéma de couplage instable, alors que celui-ci est inconditionnellement stable sans sous-cyclage. On peut récupérer la stabilité inconditionnelle si le sous-cyclage est correctement traité : l'information envoyée par le fluide au solide doit être correctement interpolée.

Le schéma CPS n'étant qu'une variante du schéma CSS, il est aussi d'ordre un en temps.

"Generalized Serial Staggered" (GSS). Dans [83] sont créées des variantes des schémas précédent dans le but d'obtenir des SCE d'ordre supérieur en temps par rapport aux schémas CSS et CPS. Pour mesurer la qualité et la stabilité des SCE proposés, l'auteur mesure les énergies envoyées de part et d'autre de l'interface entre les solveurs $\mathcal{M}_i$. Si l'interface est à l'équilibre, ces énergies doivent être exactement opposées. Sinon, un déséquilibre énergétique existe

et une énergie est créée numériquement par le SCE. Pour chaque interface Γ_{ij} entre des solveurs $\mathcal{M}_i$ et $\mathcal{M}_j$, cette énergie est notée δe_{ij}. Le SCE sera dit d'ordre m si l'on a

$$\sum_{1\leq i\leq k,\, j\in N_{i\star}} \delta e_{ij} \sim \mathcal{O}(\Delta t^m) \quad \text{quand} \quad \Delta t \to 0 \tag{2.12}$$

permettant de majorer la somme des déséquilibres énergétiques sur toutes les interfaces en fonction de Δt^m. Pour un cas simple d'un couplage à l'interface entre un fluide et une structure, l'auteur donne une estimation de l'énergie globale ΔE^N créée par le schéma à cette interface après N macro pas de temps. Elle est donnée par l'équation

$$\Delta E^N \sim N\pi(\delta e_{\mathrm{f/s}} + \delta e_{\mathrm{s/f}}) \quad \text{quand} \quad \Delta t \to 0. \tag{2.13}$$

avec $\delta e_{\mathrm{f/s}}$ et $\delta e_{\mathrm{s/f}}$ les énergies envoyées par le fluide sur la structure et par la structure sur le fluide. Dans le meilleur des cas, les énergies $\delta e_{\mathrm{f/s}}$ et $\delta e_{\mathrm{s/f}}$ se compensent au fur et à mesure des macro pas de temps et l'énergie globale reste faible. Malheureusement, ceci n'est généralement pas le cas. Les SCE créés dans [83] permettent de minimiser les énergies δe_{ij} aux interfaces entre solveurs. Ce sont des variantes du schéma CSS dans lesquelles une attention particulière est portée sur la manière dont les solveurs ont de communiquer et de transmettre leurs donnés aux autres solveurs. Avant la résolution d'un solveur, l'auteur préconise l'utilisation d'une prédiction à l'instant t^{n+1} sur les variables de couplage évaluées à t^n. La prédiction est faite à partir des dérivées temporelles des variables de couplage aux pas de temps d'intégration précédents. La forme générale du prédicteur de la variable $\mathbf{b}_{ji}$ est donnée par

$$\mathbf{b}_{ji}^{n+1,p} = \mathbf{b}_{ji}^{n} + \alpha_0 \Delta t \dot{\mathbf{b}}_{ji}^{n} + \alpha_1 \Delta t (\dot{\mathbf{b}}_{ji}^{n} - \dot{\mathbf{b}}_{ji}^{n-1}) \tag{2.14}$$

pour $\alpha_0, \alpha_1 \in \mathbb{R}$. Suite à la résolution du solveur $\mathcal{M}_i$, les variables de sorties *brutes* $\tilde{\mathbf{b}}_{ij}$ sont post-traitées de manière à obtenir les variables à transmettre $\mathbf{b}_{ij}$ aux autres solveurs. Les principaux post-traitement proposés sont donnés par

$$\mathbf{b}_{ij}^{n+1} = \tilde{\mathbf{b}}_{ij}^{n} \tag{2.15}$$

$$\mathbf{b}_{ij}^{n+1} = \tilde{\mathbf{b}}_{ij}^{n+1} \tag{2.16}$$

$$\mathbf{b}_{ij}^{n+1} = \frac{1}{\Delta t} \int_{t^n}^{t^{n+1}} \tilde{\mathbf{b}}^{ij}(t)\, \mathrm{d}t. \tag{2.17}$$

Le post-traitement est choisit en fonction de la façon dont le solveur $\mathcal{M}_i$ est résolu par le schéma de couplage. Le post-traitement donné par l'équation (2.15) correspond à un couplage explicite du solveur, celui donné par l'équation (2.16) correspond à un couplage implicite et celui donné par l'équation

(2.17) est utilisé lorsque le solveur est sous-cyclé. Suivant les choix faits pour le prédicteur (2.14) et pour la transmission des données (2.15), (2.16) et (2.17), les schémas GSS correspondant peuvent être d'ordre trois ou quatre en énergie. Le schéma CSS n'est quant à lui que d'ordre un en énergie.

Bien qu'attrayant pour les performances et la facilité de mise en œuvre, les sont parfois sujets à des problèmes de *stabilité numérique*. Ces difficultés sont connues et reconnues par l'ensemble de la communauté [84] malgré les récentes recherches et améliorations apportées aux SCE [28, 46, 31, 30, 29, 83].

Ces instabilités apparaissent notamment lorsque la force de l'interaction entre les modèles du problème est trop importante, par exemple pour l'interaction fluide structure dans [16] ou l'interaction thermique structure dans [45]. Par construction du schéma, il est peu probable que les résidus (2.11) aux interfaces entre modèles soient nuls. De ce fait, les conditions d'équilibre aux interfaces ne peuvent être assurer. Une erreur de l'ordre du macro pas de temps est introduite par le schéma de couplage à chaque macro pas de temps. Une méthode pour stabiliser ces schémas est la diminution du macro pas de temps de couplage. Toutefois, pour des raisons de performance ou parce que la taille du macro pas de temps est imposé par la physique, ceci n'est pas toujours possible. Dans ce cas et pour le problème considéré, les SCE ne peuvent plus être utilisés. Des schémas permettant de conserver un macro pas de temps suffisamment important tout en couplant fortement les modèles sont nécessaires.

2.1.7 Schémas de couplage implicites

Une manière de résoudre les problèmes de décalage en temps entre les modèles est d'utiliser un couplage implicite entre les modèles du problème de la figure 2.3. Par exemple, les équations implicites discrètes de couplage à l'interface Γ_{ij} sont données par

$$\mathbf{b}_{ij}^{n+1} = \mathcal{P}_{ij} \circ \mathcal{M}_i \left(\mathbf{b}_{ji}^{n+1}, \{\mathbf{b}_{li}^{n+1}\}_{l \in N_{\star i},\, l \neq j} \right) \tag{2.18}$$

$$\mathbf{b}_{ji}^{n+1} = \mathcal{P}_{ji} \circ \mathcal{M}_j \left(\mathbf{b}_{ij}^{n+1}, \{\mathbf{b}_{lj}^{n+1}\}_{l \in N_{\star j},\, l \neq i} \right). \tag{2.19}$$

Dans (2.18) et (2.19), à chaque macro pas de temps, le modèle $\mathcal{M}_i$ utilise les variables à t^{n+1} calculées par le modèle $\mathcal{M}_j$ tandis que le modèle $\mathcal{M}_j$ utilise les variables à t^{n+1} calculées par le modèle $\mathcal{M}_i$. De ce fait, les variables d'interfaces $\mathbf{b}_{ij}^{n+1}$ et $\mathbf{b}_{ji}^{n+1}$ respectent l'équation de point fixe (2.10) à l'interface Γ_{ij} et le résidu d'interface (2.11) est nul. L'interface est à l'équilibre et les modèles $\mathcal{M}_i$ et $\mathcal{M}_j$ sont fortement couplés.

Le processus itératif de résolution des équations de couplage aux interfaces. Cependant, bien que mathématiquement intéressantes, les équations (2.18) et

(2.19) ne permettent pas de découpler les solveurs $\mathcal{M}_i$ et $\mathcal{M}_j$. Les schémas de couplage implicites (SCI) utilisent des méthodes itératives permettant de résoudre séparément ces équations à l'intérieur du macro pas de temps. Étant donné un itéré $\mathbf{b}_{ij}^{n+1,k}$ de la variable d'interface Γ_{ij}, le SCI calcul une nouvelle variable d'interface $\tilde{\mathbf{b}}_{ij}^{n+1,k+1}$ avec les équations

$$\mathbf{b}_{ji}^{n+1,k+1} = \mathcal{P}_{ji} \circ \mathcal{M}_j \left(\mathbf{b}_{ij}^{n+1,k}, \{\mathbf{b}_{lj}^{n+1}\}_{l \in N_{\star j},\, l \neq i} \right) \tag{2.20}$$

$$\tilde{\mathbf{b}}_{ij}^{n+1,k+1} = \mathcal{P}_{ij} \circ \mathcal{M}_i \left(\mathbf{b}_{ji}^{n+1,k+1}, \{\mathbf{b}_{li}^{n+1}\}_{l \in N_{\star i},\, l \neq j} \right) \tag{2.21}$$

découplées permettant de résoudre $\mathcal{M}_i$ et $\mathcal{M}_j$ séparément. Le résidu à l'interface Γ_{ij} pour la nouvelle variable calculée est obtenu en injectant (2.20) dans (2.21)

$$\mathcal{R}_{ij}(\mathbf{b}_{ij}^{n+1,k}) = \mathcal{P}_{ij} \circ \mathcal{M}_i \left(\mathcal{P}_{ji} \circ \mathcal{M}_j \left(\mathbf{b}_{ij}^{n+1,k}, \{\mathbf{b}_{lj}\}_{l \in N_{\star j},\, l \neq i} \right), \{\mathbf{b}_{li}\}_{l \in N_{\star i},\, l \neq j} \right) - \mathbf{b}_{ij}^{n+1,k} \tag{2.22}$$

$$= \tilde{\mathbf{b}}_{ij}^{n+1,k+1} - \mathbf{b}_{ij}^{n+1,k} \tag{2.23}$$

permettant de définir un critère de convergence pour le processus itératif

$$\frac{\left\| \mathcal{R}_{ij}(\mathbf{b}_{ij}^{n+1,k}) \right\|}{\left\| \mathbf{b}_{ij}^{n+1,k} \right\|} = \frac{\left\| \tilde{\mathbf{b}}_{ij}^{n+1,k+1} - \mathbf{b}_{ij}^{n+1,k} \right\|}{\left\| \mathbf{b}_{ij}^{n+1,k} \right\|} \leq \epsilon_{\text{rel}} \tag{2.24}$$

avec ϵ_{rel} la tolérance donnée par le schéma. Si le critère de convergence (2.24) n'est pas satisfait, différentes méthodes présentées ci-après permettent le calcul d'un nouvel itéré $\mathbf{b}_{ij}^{n+1,k+1}$ de la variable d'interface Γ_{ij}. Toujours est-il qu'à convergence $k \to \infty$ du processus itératif sous-jacent au SCI à la fin du macro pas de temps, on a par construction

$$\frac{\left\| \mathcal{R}_{ij}(\mathbf{b}_{ij}^{n+1,\infty}) \right\|}{\left\| \mathbf{b}_{ij}^{n+1,\infty} \right\|} \leq \epsilon_{\text{rel}} \tag{2.25}$$

autrement dit

$$\left\| \mathcal{R}_{ij}(\mathbf{b}_{ij}^{n+1,\infty}) \right\| \leq \epsilon_{\text{rel}} \left\| \mathbf{b}_{ij}^{n+1,\infty} \right\|. \tag{2.26}$$

Le résidu à l'interface Γ_{ij} est complètement contrôlé par le schéma et peut être rendu le plus faible possible. On a alors

$$\forall i \in [1, k],\ j \in N_{i\star},\ \mathcal{R}_{ij}(\mathbf{b}_{ij}^{n+1,\infty}) \approx \mathbf{0}. \tag{2.27}$$

Les variables aux interfaces du problème vérifient leur équation de point fixe (2.10) avec une erreur majorée par la tolérance du schéma. Ce faisant, les interfaces sont à l'équilibre et un couplage fort entre modèles est obtenu.

À chaque itération, un nouvel itéré $\mathbf{b}_{ij}^{n+1,k+1}$ de la variable d'interface doit être calculé à partir de :

— l'itéré nouvellement calculé $\tilde{\mathbf{b}}_{ij}^{n+1,k}$ par les solveurs,

— et des itérés précédents $\mathbf{b}_{ij}^{n+1,l}$ pour $l \leq k$.

Cette opération correspond à un post-traitement de la variable d'interface brute $\tilde{\mathbf{b}}_{ij}^{n+1,k}$. Une solution simple est de n'effectuer aucun post-traitement, correspondant aux méthodes de *Gauss-Seidel par blocs*. Pour accélérer la convergence et la stabilité de ces méthodes, un post-traitement correspondant à une relaxation des itérés donnent naissance aux méthodes de *Gauss-Seidel par blocs avec relaxation*. Des méthodes plus complexes et plus coûteuses post-traitent les itérés avec des méthodes de Newton-Raphson menant aux méthodes dites de *Quasi-Newton*. Ces méthodes sont décrites ci-après.

La méthode de Gauss-Seidel par blocs. Aussi appelées "staggered coupling scheme" [87] ou "partitioned coupling scheme", la méthode de Gauss-Seidel par blocs résout les modèles du problème l'un après l'autre et envoie les données calculées brutes aux autres solveurs. Le nom *Gauss-Seidel* se réfère aux méthodes d'algèbre linéaire classiques et est employé pour représenter la manière dont les solveurs, les *blocs*, sont résolus et dont les données circulent entre ceux-ci. Naturellement, des méthodes de Jacobi sont aussi possibles et facilement déduites de la présentation de cette méthode. Lors de la résolution sur un macro pas de temps par une méthode de Gauss-Seidel par blocs, le nouvel itéré est donné à la fin d'une itération par

$$\mathbf{b}_{ij}^{n+1,k+1} = \tilde{\mathbf{b}}_{ij}^{n+1,k+1}. \tag{2.28}$$

L'algorithme 3 donne les étapes de la résolution du problème de couplage par le schéma SCI de Gauss-Seidel par blocs de l'instant initial de simulation t^0 à l'instant final t^{n_f}.

Cependant, le SCI Gauss-Seidel par blocs ne fait qu'itérer à chaque macro pas de temps le SCE CSS précédent. Par conséquent, les problèmes de stabilités numériques inhérent à ce schéma persiste à chaque itération de la méthode. Pour stabiliser la méthode de Gauss-Seidel par blocs, des techniques de relaxation sont utilisées.

La méthode de Gauss-Seidel par blocs avec relaxation. Les itérés brutes des variables d'interfaces sont post-traités par des techniques de relaxation

Algorithme 3 Algorithme de résolution du SCI de Gauss-Seidel.

1: **for** $n = 0$ to $n_f - 1$ **do**
2: **while** non convergence **do**
3: **for** $i = 0$ to k **do**
4: $\{\mathbf{b}_{il}^{n+1,k+1}\}_{l \in N_{i\star}} = \mathcal{M}_i(\{\mathbf{b}_{ji}^{n+1,k+1}\}_{j<i,\, j \in N_{\star i}}, \{\mathbf{b}_{ji}^{n+1,k}\}_{j>i,\, j \in N_{\star i}})$.
5: **for** $l \in N_{i\star}$ **do**
6: mesurer la convergence entre $\mathbf{b}_{il}^{n+1,k+1}$ et $\mathbf{b}_{il}^{n+1,k}$.
7: envoyer $\mathbf{b}_{il}^{n+1}$ à $\mathcal{M}_l$.
8: **end for**
9: **end for**
10: **end while**
11: **end for**

données par l'équation de relaxation

$$\mathbf{b}_{ij}^{n+1,k+1} = \mathbf{b}_{ij}^{n+1,k} + \omega^k \mathcal{R}_{ij}(\mathbf{b}_{ij}^{n+1,k}) \tag{2.29}$$

$$= \mathbf{b}_{ij}^{n+1,k} + \omega^k (\tilde{\mathbf{b}}_{ij}^{n+1,k+1} - \mathbf{b}_{ij}^{n+1,k}) \tag{2.30}$$

$$= (1 - \omega^k)\, \mathbf{b}_{ij}^{n+1,k} + \omega^k\, \tilde{\mathbf{b}}_{ij}^{n+1,k+1} \tag{2.31}$$

avec ω^k le paramètre de relaxation, dynamique ou non ($\omega^k < 1$ correspondant à de la sous-relaxation et $\omega^k > 1$ correspondant à de la sur-relaxation). L'algorithme 4 donne les étapes de la résolution du problème de couplage par le schéma SCI de Gauss-Seidel par blocs de l'instant initial de simulation t^0 à l'instant final t^{n_f}. Différentes méthodes sont disponibles pour la relaxation de $\mathbf{b}_{il}^{n+1,k+1}$ à la ligne 9. Elles seront présentées par la suite.

Les méthodes de Quasi-Newton. Partant de l'équation de résidu (2.22) donnant $\mathcal{R}_{ij}(\mathbf{b}_{ij}^{n+1,k})$, le nouvel itéré est calculé en suivant la direction donnée par des méthodes de Newton-Raphson pour la résolution du problème $\mathcal{R}_{ij}(\mathbf{b}_{ij}) = \mathbf{0}$. Il est donné par l'équation

$$\mathbf{b}_{ij}^{n+1,k+1} = \mathbf{b}_{ij}^{n+1,k} + \left(\frac{\mathrm{d}\mathcal{R}_{ij}}{\mathrm{d}\mathbf{b}_{ij}}\Big|_{\mathbf{b}_{ij}^{n+1,k}} \right)^{-1} \left[-\mathcal{R}_{ij}(\mathbf{b}_{ij}^{n+1,k} \right]. \tag{2.32}$$

Le calcul de (2.32) nécessite la connaissance des matrices jacobiennes des résidus aux interfaces. Si celles-ci sont connues, une résolution directe peut être faite. Cependant, celles-ci sont généralement inconnues et/ou difficilement calculables puisque les solveurs sont considérés comme des boîtes noires dont on ne peut potentiellement extraire que peu d'information (c'est par exemple le cas dans le contexte AG). Par conséquent, les itérations ne peuvent qu'être approximées en approchant la matrice jacobienne ou son inverse : ceci mène

Algorithme 4 Algorithme de résolution du SCI de Gauss-Seidel par blocs

for $n = 0$ to $n_f - 1$ **do**
  while non convergence **do**
    for $i = 0$ to k **do**
      $\{\mathbf{b}_{il}^{n+1,k+1}\}_{l \in N_{i\star}} = \mathcal{M}_i(\{\mathbf{b}_{ji}^{n+1,k+1}\}_{j<i,\, j \in N_{\star i}}, \{\mathbf{b}_{ji}^{n+1,k}\}_{j>i,\, j \in N_{\star i}})$.
      for $l \in N_{i\star}$ **do**
        if $\mathcal{M}_l$ déjà résolu **then**
          mesurer la convergence entre $\tilde{\mathbf{b}}_{il}^{n+1,k+1}$ et $\mathbf{b}_{il}^{n+1,k}$.
          if non convergence **then**
            calculer $\mathbf{b}_{il}^{n+1,k+1}$ par relaxation.
          end if
        else
          $\mathbf{b}_{il}^{n+1,k+1} = \tilde{\mathbf{b}}_{il}^{n+1,k+1}$
        end if
        envoyer $\mathbf{b}_{il}^{n+1,k+1}$ à $\mathcal{M}_l$.
      end for
    end for
  end while
end for

aux méthodes de *Quasi-Newton*. Différentes méthodes existent suivant la manière d'approcher ces matrices. Dans [44], Gerbeau et al. utilisent des modèles d'ordres réduits pour calculer la matrice jacobienne. Dans [77], Michlet et al. utilisent des itérations de Newton-Krylov à l'intérieur du processus itératif pour calculer une approximation de la matrice jacobienne, menant aux méthodes de Quasi-Newton appelées "*Jacobian free Newton-Krylov*". À l'intérieur d'un macro pas de temps et depuis les valeurs des résidus des itérations précédentes, Degroote et al. dans [21] résolvent des problèmes de moindres carrés pour approcher l'inverse de la matrice jacobienne : on parle alors de méthodes "*Interface Quasi-Newton - Inverse Least-Squares*" (IQN-ILS). Vierendeels et al. [94] utilisent des techniques similaires pour approximer cette fois directement la matrice jacobienne, ces méthodes sont donc appelées "*Interface Quasi-Newton - Least-Squares*" (IQN-LS). Des algorithmes de Quasi-Newton basés sur des méthodes multi-grilles sont utilisés dans [22]. Une revue des différentes méthodes de Quasi-Newton appliquées à des problèmes d'interaction fluide structure peut être trouvée dans [78]. Une revue de ces méthodes appliquées aux problèmes d'interaction thermique structure peut être trouvée dans [41].

Un point commun de ces méthodes de Quasi-Newton est le coût en temps et en stockage qui peut devenir important. La reconstruction de la matrice jacobienne ou de son inverse par les différentes méthodes peut être un processus long et difficile. Comme souligné dans [64], les méthodes de Gauss-

Seidel par blocs avec relaxation peuvent être vues comme des méthodes de Quasi-Newton pour lesquelles la reconstruction de ces matrices est donnée par

$$\left(\frac{\mathrm{d}\mathcal{R}_{ij}}{\mathrm{d}\mathbf{b}_{ij}}\Big|_{\mathbf{b}_{ij}^{n+1,k}}\right)^{-1} \approx -\omega^k\, I. \tag{2.33}$$

En effet, en ré-injectant (2.33) dans l'équation (2.32) de post-traitement de la méthode de Quasi-Newton, il vient

$$\mathbf{b}_{ij}^{n+1,k+1} = \mathbf{b}_{ij}^{n+1,k} + -\omega^k\, I\, \left[-\mathcal{R}_{ij}(\mathbf{b}_{ij}^{n+1,k})\right] \tag{2.34}$$

on retrouve bien l'équation (2.29) de post-traitement de la méthode de Gauss-Seidel par blocs avec relaxation.

Cependant, par rapport aux méthodes de Quasi-Newton du paragraphe précédent, le coût de la reconstruction de la matrice jacobienne dans (2.33) est moindre pour la méthode de Gauss-Seidel par blocs avec relaxation. Il est montré dans [66] que la méthode de Gauss-Seidel par blocs avec des techniques de relaxation se révèle très efficace avec un coût faible en comparaison aux méthodes de Quasi-Newton présentées précédemment. De plus, il est montré dans [86] que les méthodes de Gauss-Seidel par blocs avec relaxation s'avèrent très intéressantes en comparaison à la résolution direct des itérés de Newton (2.32) avec la matrice jacobienne exacte.

Par conséquent, *on se concentre par la suite sur les SCI utilisant des méthodes de Gauss-Seidel par blocs avec relaxation*.

Ci-après sont présentées différentes techniques de calcul du paramètre de relaxation ω^k. Les techniques de relaxation *constante* et par *méthode de la sécante* sont celles retenues pour cette thèse (elles sont présentées par la suite). Beaucoup d'autres techniques existent, par exemple la relaxation par méthode de la plus profonde descente dans [66]. Une revue complète des méthodes de relaxation vectorielle peut être trouvée dans [86]. Notamment des méthodes d'ordre deux comme la méthode de Steffensen accélérée par un procédé Δ^2 d'Aitken [1]. Ces méthodes sont par exemple appliquées dans [8].

Relaxation constante. Naturellement, une méthode simple et très peu coûteuse est de prendre un paramètre de relaxation $\omega^k = \omega$ constant. Si

$$0 < \omega < 1,$$

les itérés sont sous-relaxés et la relaxation stabilise les itérations de couplage au détriment de la vitesse de convergence des itérations. Si

$$1 \leq \omega < 2,$$

les itérés sont sur-relaxés et la relaxation accélère les itérations de couplage au détriment de la stabilisation.

Cependant, la valeur à prendre pour obtenir le taux de convergence optimal des itérés est fortement dépendante du problème considéré (voir [86] par exemple). La recherche du paramètre optimal pour un problème se fait généralement par des tests et peut s'avérer laborieuse. Des techniques calculant automatiquement ce paramètre sont donc préférables, par exemple la *relaxation dynamique par méthode de la sécante*.

Relaxation dynamique par méthode de la sécante. Dans cette méthode, la technique de relaxation se base sur la résolution de l'équation

$$\mathcal{R}_{ij}(\mathbf{b}_{ij}) = \mathbf{0} \tag{2.35}$$

par la méthode itérative de la sécante, voir [56] par exemple. Dans le cas scalaire, les itérés de la méthode sont donnés par

$$b_{ij}^{n+1,k+1} = b_{ij}^{n+1,k} - \frac{b_{ij}^{n+1,k} - b_{ij}^{n+1,k-1}}{\mathcal{R}_{ij}(b_{ij}^{n+1,k}) - \mathcal{R}_{ij}(b_{ij}^{n+1,k-1})} \mathcal{R}_{ij}(b_{ij}^{n+1,k}). \tag{2.36}$$

On peut déduire de (2.36) et de l'équation de relaxation (2.29) scalaire

$$b_{ij}^{n+1,k+1} = b_{ij}^{n+1,k} + \omega^k \, \mathcal{R}_{ij}(b_{ij}^{n+1,k}), \tag{2.37}$$

la valeur du paramètre de relaxation dynamique ω^k. Il est donné par l'équation

$$\omega^k = -\frac{b_{ij}^{n+1,k} - b_{ij}^{n+1,k-1}}{\mathcal{R}_{ij}(b_{ij}^{n+1,k}) - \mathcal{R}_{ij}(b_{ij}^{n+1,k-1})}. \tag{2.38}$$

Puis en injectant récursivement dans (2.38) la valeur de ω^{k-1} donnée par l'équation de relaxation scalaire (2.37), il vient

$$\omega^k = -\omega^{k-1} \frac{\mathcal{R}_{ij}(b_{ij}^{n+1,k-1})}{\mathcal{R}_{ij}(b_{ij}^{n+1,k}) - \mathcal{R}_{ij}(b_{ij}^{n+1,k-1})}. \tag{2.39}$$

Pour des cas non-scalaires, [53] suggère d'étendre la cas scalaire (2.39) au cas vectoriel par

$$\omega^k = -\omega^{k-1} \frac{\langle \mathcal{R}_{ij}(\mathbf{b}_{ij}^{n+1,k-1}), \mathcal{R}_{ij}(\mathbf{b}_{ij}^{n+1,k}) - \mathcal{R}_{ij}(\mathbf{b}_{ij}^{n+1,k-1}) \rangle}{\left\| \mathcal{R}_{ij}(\mathbf{b}_{ij}^{n+1,k}) - \mathcal{R}_{ij}(\mathbf{b}_{ij}^{n+1,k-1}) \right\|^2}. \tag{2.40}$$

Les itérés de la méthode de la sécante convergent vers la solution de l'équation (2.10) de point fixe à l'interface, ou de manière équivalente vers la solution de l'équation du résidu $\mathcal{R}_{ij}(\mathbf{b}_{ij}) = \mathbf{0}$, avec un taux de convergence de

$\frac{1+\sqrt{5}}{2}$. La méthode est d'ordre moins élevé que d'autres méthodes dans [86] (certaines méthodes sont d'ordre 2 par exemple). Cependant, pour chaque itération à l'intérieur du macro pas de temps, les méthodes proposées dans [86] requièrent plusieurs appels de l'opérateur de résidu pour le calcul du nouvel itéré. Tandis que pour les itérés de la méthode de la sécante donnés par (2.40), seule une évaluation de l'opérateur de résidu par itération est nécessaire. Pour l'itéré initial ω^0 du paramètre de relaxation au début de la résolution sur un macro pas de temps au temps t^n, il est suggérer dans [66] de choisir la valeur par l'équation

$$\omega^0 = \max\left(\omega^{n-1,\infty}, \omega_{\max}\right)$$

avec $\omega^{n-1,\infty}$ la valeur obtenue a convergence au macro pas de temps précédent et $\omega_{\max}$ un paramètre fixe du problème.

2.1.8 Choix du schéma de couplage

Ont été présentés dans les sections précédentes les SCE et SCI, respectivement schéma de couplage explicite et implicite. permettant de résoudre le couplage entre modèles. Le couplage obtenu à la fin du macro pas de temps est plus ou moins fort suivant le choix du schéma. La nature du problème physique permet généralement de déterminer le type de schéma à choisir. Si la force de l'interaction entre modèles est trop importante, un SCI est conseillé. Une telle force impose des conditions de stabilité qui peuvent être contraignantes (notamment sur la taille du macro pas de temps) pour qu'un SCE puisse être utilisé en gardant des performances de calcul raisonnable (c'est à dire avec un macro pas de temps très faible). Dans certain cas, le SCE peut même être instable peu importe le macro pas de temps. C'est par exemple le cas dans [16] pour un couplage fluide structure lorsque

$$\frac{\rho_S h_S}{\rho_F \mu_{max}} < 1$$

avec ρ_S la densité du solide, ρ_L la densité du fluide, h_S une constante dépendant des paramètres physiques du problème et μ_{max} un constante dépendant des opérateurs mathématiques de discrétisation utilisés. Le SCI de Gauss-Seidel par blocs avec relaxation permet de résoudre le couplage de manière stable en conservant un macro pas de temps suffisamment large. Une condition assurant la convergence des itérations du SCI y est même calculé. Pour des macro pas de temps petits, elle est donnée par

$$0 < \omega < 2/(1 + \frac{\rho_F \mu_{max}}{\rho_S h_S}). \tag{2.41}$$

Notons que dans la zone d'instabilité du SCE, c'est à dire $\rho_S h_S / \rho_F \mu_{max} < 1$, le paramètre de relaxation ω du SCI doit vérifier $0 < \omega < 1$ correspondant à une sous-relaxation.

Il est souligné dans [16] que de tels conditions dépendent, certes d'une part du problème physique sous-jacent, mais aussi de la manière de coupler les modèles du problème partitionné. Bien que certaines conditions doivent être vérifiées pour assurer que le problème de décomposition de domaine associé au problème partitionné soit bien posé [24], des degrés de liberté persistent sur le choix du couplage. Par exemple dans [16], un couplage fluide structure de type Dirichlet-Neumann donne la condition de convergence donnée par l'équation (2.41) alors qu'un couplage Neumann-Dirichlet donne

$$0 < \omega < 2/(1 + \frac{\rho_S h_S}{\rho_F \mu_{max}})$$

renversant le ratio des densités (voir [24] pour la terminologie Dirichlet-Neumann empruntée de la décomposition de domaine). Des calculs similaires de stabilité pour des couplages Dirichlet-Neumann ou Neumann-Dirichlet sur des problèmes de thermique peuvent être trouvés dans [45].

De manière générale, la stabilité de la résolution du couplage dépend d'une part du *problème physique considéré* et d'autre part du *partitionnement* effectué du problème.

Néanmoins, les schémas introduits ne permettent pas, jusqu'à présent, de synchroniser les modèles sur des événements pouvant survenir à l'intérieur de la macro boucle en temps. Ci-après est proposé un *schéma de synchronisation des modèles*. Ce schéma permet de calculer au cours des itérations de résolution du couplage le macro pas de temps Δt permettant la synchronisation des modèles sur un événement.

2.2 Méthodes de synchronisation des modèles

Les modèles couplés du système complexe de la figure 2.4 peuvent avoir différents états internes. La transition d'un état vers un autre d'un modèle est déclenchée par des événements internes à celui-ci correspondant à l'activation d'une fonction de seuil. L'activation ou non de la fonction est dépendante des variables d'entrées envoyées par les modèles couplés. Par conséquent, les temps auquel vont se déclencher les événements sont des inconnus du couplage.

2.2.1 L'importance de la détection des événements

Comme décrit à la figure 2.7, chaque état à son propre solveur, ses propres équations et ses propres interfaces. De ce fait, les transitions d'états peuvent engendrer des discontinuités dans les variables d'états des modèles et/ou dans les variables d'interfaces. Lors de ces transitions, les fonctions mathématiques du problème de couplage (2.5) peuvent être discontinues amenant

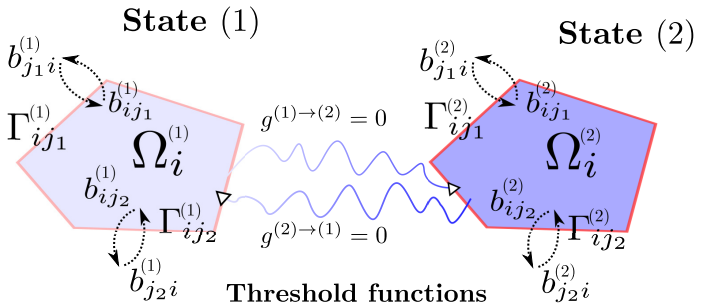


FIGURE 2.7 – Représentation schématique des états $\Omega_i^{(1)}$ et $\Omega_i^{(2)}$ du domaine Ω_i. Chaque état a ses propres interfaces $\Gamma_\bullet^{(1)}$ et $\Gamma_\bullet^{(2)}$, et ses propres variables d'interface $\mathbf{b}_\bullet^{(1)}$ et $\mathbf{b}_\bullet^{(2)}$. Une transition d'état est déclenchée quand une fonction de seuil traverse zéro.

les théorèmes usuels d'intégration numérique en dehors de leur domaine de validité, par exemple la non convergence des itérations d'un SCI (un exemple de non convergence des itérations d'un SCI lors d'une discontinuité sur une application industrielle de la plate-forme PROCOR est donnée dans la section 4.3).

De plus, si le schéma ne parvient pas à se synchroniser sur les événements à l'intérieur de la macro boucle en temps, il peut être amener à louper des événements majeurs et critiques. Par exemple la disparition d'un modèle modifiant la topologie du graphe de la figure 2.3 de couplage du problème. Le système complexe peut alors se retrouver dans un état physique incohérent.

Par conséquent, pour assurer le cohérence physique du système à tout instant de la résolution et le bon comportement des algorithmes numériques de résolution, la détection des événements et la synchronisation des modèles est primordiale. Toutefois, dans la contexte de l'approche partitionnée, il n'y a à notre connaissance que peu d'algorithmes permettant cette synchronisation. Néanmoins, de tels algorithmes existent dans le contexte des équations différentielles algébriques [82] dans lequel des événements similaires peuvent être déclenchés [74, 81, 72].

2.2.2 Objectif de la détection des événements

À l'intérieur de la macro boucle en temps, le schéma de couplage doit être capable d'adapter son macro pas de temps pour synchroniser tous les modèles sur le temps du premier événement survenu. Cependant, les stratégies de synchronisation sont difficiles à mettre en place avec les SCE. Au mieux, ceux-ci ne peuvent que synchroniser les modèles à chaque instant de la macro boucle en temps. L'erreur de détection est donc de l'ordre du macro pas de temps Δt. Ci-après, on décrit comment un SCI a pu être adapté pour la

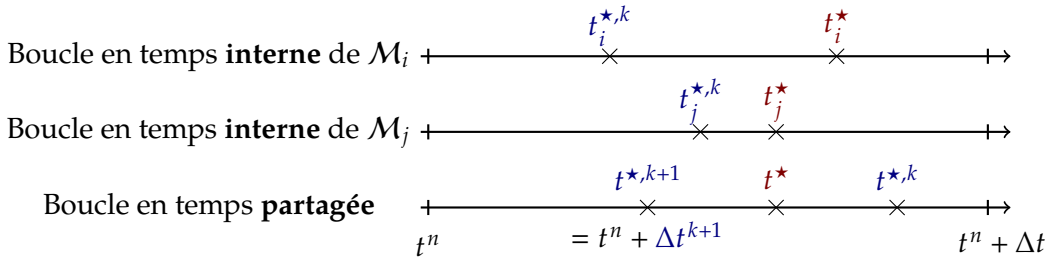


FIGURE 2.8 – La boucle en temps partagée par les solveurs $\mathcal{M}_i$ et $\mathcal{M}_j$ et leur boucle en temps interne. Les événements internes aux solveurs sont déclenchés aux temps $t_i^\star$ et $t_j^\star$ et doivent être détectés pour synchroniser les modèles à l'instant $t^\star$ du premier événement. Durant ses itérations entre l'instant t^n et $t^{\star,k}$, un SCI détecte les événements aux temps $t_i^{\star,k}$ et $t_j^{\star,k}$, et doit calculer le nouvel itéré $t^{\star,k+1}$ tel que $t^{\star,\infty} \approx t^\star$.

détection d'événement pour assurer la synchronisation entre les modèles.

Considérons le couplage à l'interface Γ_{ij}. Comme décrit à la figure 2.8, des événements sont déclenchés à l'intérieur de la macro boucle en temps Δt par le solveur $\mathcal{M}_i$ et/ou le solveur $\mathcal{M}_j$ à des temps $t_i^\star$ et $t_j^\star$ entre t^n et $t^n + \Delta t$. Les solveurs devraient être synchronisés à l'instant auquel apparaît le premier événement, ici au temps $t^\star = t_j^\star$.

Lorsque des événements sont déclenchés par les solveurs $\mathcal{M}_i$ et $\mathcal{M}_j$ à l'intérieur d'un macro pas de temps, le temps final atteint par un SCI, correspondant au temps du premier événement, change et est calculé au cours des itérations. De ce fait, il est noté $t^{\star,k}$. Pour la première itération $k = 0$, on prendra

$$t^{\star,k=0} = t^{n+1} = t^n + \Delta t^{k=0} = t^n + \Delta t\,. \tag{2.42}$$

Lorsque les événements sont déclenchés, les solveurs arrêtent leur simulation avant la fin du pas de temps. Ces événements apparaissent aux instants $t_i^{\star,k}$ et $t_j^{\star,k}$ et les macro pas de temps calculés par les deux solveurs sont Δt_i^k et Δt_j^k. Il faut ensuite calculer le nouveau macro pas de temps Δt^{k+1} donnant le temps de l'événement à l'itération $k+1$ par

$$t^{\star,k+1} = t^n + \Delta t^{k+1}. \tag{2.43}$$

À convergence $k = \infty$ des itérations du SCI, le but est naturellement d'avoir

$$t^{\star,\infty} \approx t^\star. \tag{2.44}$$

2.2.3 Algorithme de détection des événements

L'algorithme 5 donne les étapes permettant le calcul du nouvel itéré $t^{\star,k+1}$. Il est utilisé à la fin d'une itération d'un SCI entre l'instant t^n et l'instant $t^{\star,k} =$

Algorithme 5 Calcul du nouvel itéré $t^{\star,k+1}$ à la fin d'une itération d'un SCI entre l'instant t^n et $t^{\star,k} = t^n + \Delta t^k$

1: $\Delta t_i^k \leftarrow$ le macro pas de temps calculé par $\mathcal{M}_i$
2: $\Delta t_j^k \leftarrow$ le macro pas de temps calculé par $\mathcal{M}_j$
3: $\tilde{\Delta t}^k \leftarrow \min(\Delta t_i^k, \Delta t_j^k)$
4: **if** $\tilde{\Delta t}^k < \Delta t^k$ **then**
5: **if** $(|\tilde{\Delta t}^k - \Delta t^k| / \tilde{\Delta t}^k < \epsilon_{\text{rel}}^{\Delta t}$ **then**
6: $\Delta t^{k+1} \leftarrow \tilde{\Delta t}^k$
7: **else**
8: $\Delta t^{k+1} \leftarrow \alpha \tilde{\Delta t}^k + (1-\alpha)\Delta t^k$
9: **end if**
10: **else if** $\tilde{\Delta t}^k < \Delta t$ **then**
11: $\Delta t^{k+1} \leftarrow \beta \tilde{\Delta t}^k + (1-\beta)\Delta t$
12: **end if**

$t^n + \Delta t^k$.

De la ligne 1 à 3, l'algorithme récupère des solveurs le macro pas de temps qu'ils ont été capables de calculer (correspondant à l'apparition d'un événement s'il est plus petit que le macro pas de temps cible Δt^k). Puis, il calcule le minimum de ces pas de temps correspondant au premier événement détecté.

Ensuite, deux cas se présentent. Soit un événement est apparu entre l'instant t^n et $t^n + \Delta t^k$, c'est à dire

$$\tilde{\Delta t}^k < \Delta t^k, \tag{2.45}$$

soit aucun événement n'est apparu au cours de l'itération mais un événement avait été détecté au cours des itérations précédentes, c'est à dire

$$\tilde{\Delta t}^k == \Delta t^k \quad \wedge \quad \tilde{\Delta t}^k < \Delta t. \tag{2.46}$$

Dans le premier cas (ligne 4 et équation (2.45)), la convergence du macro pas de temps calculé $\tilde{\Delta t}^k$ par rapport au macro pas de temps de l'itération précédente Δt^k est testée à la ligne 5. Le critère de convergence $\epsilon_{\text{rel}}^{\Delta t}$ est donné par le schéma et n'est pas forcément le même que le critère de convergence ϵ_{rel} utilisé précédemment pour tester la convergence aux interfaces entre solveurs. Si il y a convergence, ligne 6, le nouveau macro pas de temps est pris tel

quel. Si le pas de temps n'a pas convergé, le macro pas de temps Δt^{k+1} pour l'itération $k+1$ est donné par l'équation de relaxation ligne 8. La relaxation a un signification particulière pour la détection de l'événement :

— Si une relaxation trop faible est appliquée sur le macro pas de temps, c'est à dire $\alpha \approx 0$ et $\Delta t^{k+1} \approx \Delta t^k$, les modèles ne seront pas conscients de l'événement lors de l'itération suivante du SCI. Ceux-ci seront donc résolus sur un pas de temps similaire au pas de temps précédent et le couplage entre les modèles n'évoluera pas ou évoluera très lentement. Le temps de détection de l'événement ne changera pas ou changera très lentement et restera proche de celui calculé lors de l'itération k. Du fait de la *forte sous relaxation*, la convergence vers le véritable temps de l'événement sera très lente, mais aussi très stable.

— À l'inverse, si une relaxation trop importante est appliquée sur le macro pas de temps, c'est à dire $\alpha \approx 1$ et $\Delta t^{k+1} \approx \tilde{\Delta t}^k$, les modèles verront directement à l'itération suivante un événement à l'instant $t^n + \Delta t^{k+1} \approx t^n + \tilde{\Delta t}^k$ et seront résolus sur ce macro pas de temps. De ce fait, le couplage va évoluer très rapidement et le changement par rapport à l'itération précédente peut être brutal, par exemple si $\Delta t^{k+1} \ll \Delta t^k$. Dans ce cas, l'événement peut ne plus être détecté à l'itération $k+1$ parce qu'il apparaît après l'instant $t^n + \Delta t^{k+1}$ et l'algorithme passera par la ligne 10, décrit ci-après.

Dans le second cas (ligne 10 **et équation** (2.46)**),** un événement avait été détecté lors de précédentes itérations, l'algorithme avait alors réduit le macro pas de temps de couplage Δt^k. Ceci a modifié le couplage entre modèles pour l'itération k ce qui a reculé l'instant de l'événement à un instant postérieur à $t^n + \Delta t^k$. Dans ce cas, le nouveau macro pas de temps $t^n + \Delta t^{k+1}$ est donné par l'équation de relaxation ligne 11 utilisée car :

— Si $\Delta t^{k+1} \approx \Delta t$, alors toutes les informations acquises au cours des itérations précédentes sont perdues et l'algorithme bouclera à l'infini.

— Si $\Delta t^{k+1} \approx \tilde{\Delta t}^k$, l'événement ne sera plus détecté.

L'équation de relaxation permet donc de relancer l'algorithme de détection sans perdre toutes les informations acquises au cours des itérations précédentes.

Les valeurs de α et β sont pour le moment fixes et données à la création du SCI muni de l'algorithme de détection. Une relaxation dynamique s'adaptant à la dynamique d'apparition de l'événement est possible et est en cours de test.

À convergence de l'algorithme (5), le couplage entre modèles a été résolu entre les temps t^n et $t^{\star,\infty} = t^n + \Delta t^\infty$. Le prochain macro pas de temps démarre à l'instant t^{n+1} avec le même algorithme.

2.2.4 Couplage obtenu à convergence d'un SCI muni de l'algorithme de synchronisation

À convergence $k = \infty$ entre les instants t^n et $t^{n+1} = t^{\star,\infty}$ d'un SCI muni de l'algorithme de synchronisation à l'interface Γ_{ij} entre $\mathcal{M}_i$ et $\mathcal{M}_j$, on a que :

— Le modèle i et le modèle j sont fortement couplés car l'équation de point fixe (2.10) est vérifié avec une erreur contrôlée par la tolérance ϵ_{rel} du SCI, c'est à dire :

$$\frac{\left\| \mathcal{M}_j^{\Delta t} \circ \mathcal{M}_i^{\Delta t}(\mathbf{b}_{ji}^{n+1,\infty}) - \mathbf{b}_{ji}^{n+1,\infty} \right\|}{\left\| \mathbf{b}_{ji}^{n+1,\infty} \right\|} \leq \epsilon_{\text{rel}}. \tag{2.47}$$

— Le modèle i et le modèle j sont synchronisés sur de potentiels événements internes aux modèles. En particulier, ils sont synchronisés sur l'instant du premier événement avec un erreur contrôlée par la tolérance $\epsilon_{\text{rel}}^{\Delta t}$, c'est à dire :

$$\frac{\left| t^{n+1} - t^{\star} \right|}{\Delta t^{\infty}} = \frac{\left| t^{\star,\infty} - t^{\star} \right|}{\Delta t^{\infty}} \leq \epsilon_{\text{rel}}^{\Delta t}. \tag{2.48}$$

Les modèles sont fortement couplés à la fin du macro pas de temps et synchronisés sur l'événement potentiellement associé à une forte discontinuité. Les modèles peuvent ensuite être résolus sur les pas de temps suivant en repartant du temps de l'événement. Les erreurs créées lorsque les modèles ne sont pas synchronisés sur la discontinuité, se traduisant par de fortes erreurs numériques ou des états physiques incohérents des modèles, sont ainsi évitées (des exemples sur des résultats numériques de ces erreurs sont donnés dans les sections 4.2 et 4.3).

Des exemples d'utilisation d'un SCI muni de l'algorithme de synchronisation et des applications numériques sont donnés dans les sections 4.1, 4.2 et 4.3 du chapitre 4. Au travers de ces sections, on met en avant l'impact positif qu'a l'utilisation de tels schémas de couplage et de synchronisation sur

— *la précision et la stabilité* des calculs issus de la plate-forme PROCOR.

— *la structure et la propreté* du code logiciel de la plate-forme PROCOR.

Néanmoins, avant d'utiliser les schémas de couplage présentés et l'algorithme de synchronisation, il faut pouvoir les insérer dans la plate-forme. Dans le chapitre suivant 3, on présente l'architecture logicielle mise en place permettant l'implémentation des différents algorithmes de couplage et de synchronisation.

Chapitre 3

Architecture logicielle pour le couplage et la synchronisation

Le contexte AG est décrit dans la section 1.1 et le cycle de vie (décrit à la figure 1.8) qu'il impose aux applications industrielles de la plate-forme PROCOR est détaillé dans la section 1.2. Ce cycle de vie impose une forte évolution de la modélisation au sein d'une application et, de ce fait, impose un découpage de l'application en blocs de modèles. Chaque bloc étant vu par l'application comme une *boîte noire* avec des entrées, des sorties et représentant une partie du réacteur ou un phénomène physique. L'approche partitionnée (détaillée dans la section 2.1), particulièrement adaptée à cette vue par blocs de modèles, propose des schémas de résolution du couplage des modèles. Dans la section 2.2, on a décrit un algorithme permettant la synchronisation des modèles couplés.

Reste l'implémentation logicielle de ces solutions de couplage et de synchronisation dans la plate-forme logicielle CEA PROCOR [69], sujet de ce chapitre.

Les outils et les méthodes utilisés pour la conception de l'architecture logicielle de couplage et de synchronisation sont ceux trouvés dans [85]. En particulier, la conception a suivi le cycle illustré à la figure 3.1 (cycle en *V*). L'abscisse de ce cycle correspond au temps dans le processus de conception tandis que l'ordonnée correspond au niveau de détail de l'étape de conception. Plus l'étape est haute, moins elle sera détaillée et plus elle aura une vue d'ensemble sur l'architecture. À l'inverse, plus elle sera basse, plus elle rentrera dans les détails de la conception de l'architecture.

Les sections de ce chapitre suivent l'ordre chronologique du cycle, et donc son niveau de détail :

— La section 3.1 décrit l'**analyse des besoins** de la plate-forme et de ses utilisateurs de l'architecture de couplage par des cas d'utilisation. Puis

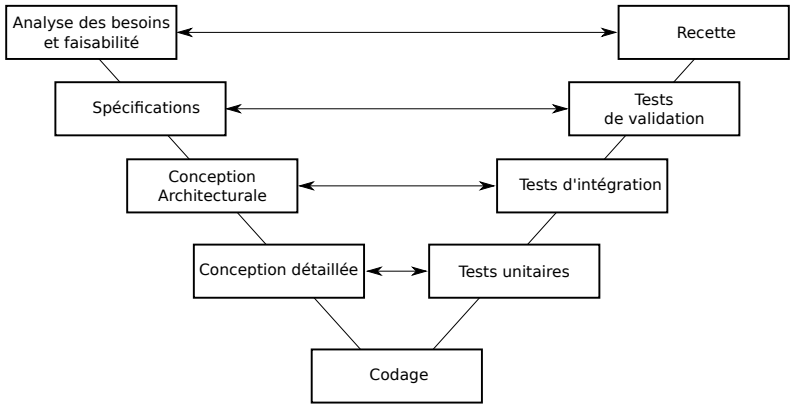


FIGURE 3.1 – Cycle de développement suivi pour la création de l'architecture de couplage et de synchronisation des modèles de la plate-forme PROCOR (cycle en *V*).

sont détaillées les spécifications de l'architecture en matière d'*exigences fonctionnelles* et d'*exigences non-fonctionnels*.

— La section 3.2 décrit la plate-forme PROCOR avant l'intégration de la nouvelle architecture permettant ainsi de comprendre certains choix faits lors de sa conception et de comprendre le fonctionnement de certains classes de la plate-forme originelle réutilisées par la suite.

— La section 3.3 décrit la **conception** de l'architecture logicielle, généralement au travers de diagramme UML, ainsi que les différents algorithmes de couplage implémentés.

La définition des besoins de l'architecture de couplage, sa conception logicielle et son intégration dans la plate-forme PROCOR se sont faites sous la contrainte de *non-régression* imposée par le contexte industrielle de la plate-forme PROCOR. Lorsqu'une application, un modèle ou n'importe quelle classe de l'architecture originelle associés à des résultats industriels sont modifiés, il faut assurer que, à conditions identiques (par exemple même schéma de résolution du couplage), les résultats sont conservés sous la nouvelle architecture.

3.1 Le besoin logiciel de l'architecture de couplage et de synchronisation

3.1.1 Analyse du besoin et des objectifs principaux de l'architecture de couplage

L'architecture doit résoudre les problématiques de couplage et de synchronisation exprimées depuis le début de ce manuscrit. Des problématiques décrites depuis la début du manuscrit ont été identifiées trois parties différentes :

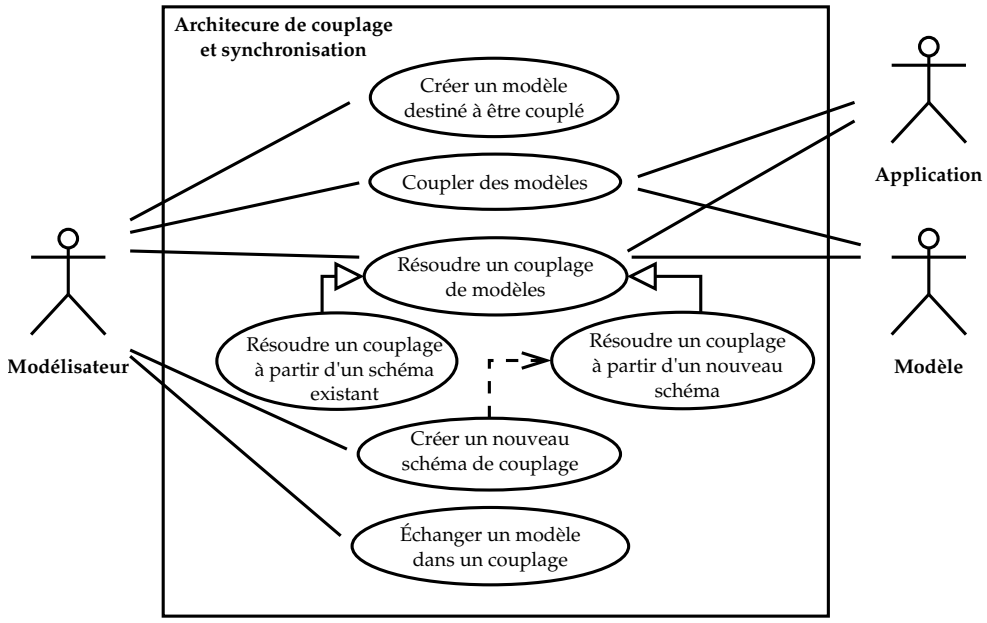


FIGURE 3.2 – Cas d'utilisation de l'architecture de couplage et synchronisation des modèles.

— Les applications PROCOR composées d'un ensemble de modèles couplés.
— Les modèles potentiellement composés de sous modèles couplés.
— Les développeurs et/ou modélisateurs de la plate-forme.

L'architecture doit répondre aux besoins des différentes parties interagissant avec elle. Les trois parties ont des utilisations et des interactions différentes avec l'architecture de couplage et de synchronisation. Celles-ci sont décrites dans le diagramme de cas d'utilisation de la figure 3.2. Le modélisateur veut créer des modèles destinés à être couplés dans une application ou en tant que sous-modèle d'un modèle. Ce couplage doit être résolu en utilisant un des schémas de couplage et de synchronisation décrits précédemment ou à partir d'un nouveau schéma créé par le modélisateur. Enfin, l'application, le modèle ou le modélisateur doivent pouvoir aisément échanger un modèle dans un couplage du fait du fort potentiel d'évolution des modèles dans le contexte AG ou pour des tests numériques ou de modélisation.

La description des interactions qu'ont les différentes parties avec l'architecture permet d'identifier les objectifs principaux de l'architecture :
— Facilité la définition, l'évolution et la résolution du couplage des modèles.
— Facilité la création ou la modification des modèles au sein du couplage.
— Facilité la création de nouveau schéma de résolution du couplage et de synchronisation du fait des deux points précédents qui peuvent faire

évoluer le couplage et donc le besoin en termes de schéma numérique de résolution.

La définition et la résolution du couplage et la synchronisation des modèles. L'architecture permet aux applications, aux modèles ou aux modélisateurs de définir un couplage entre différents modèles. Les modèles doivent être considérés comme des boîtes noires et doivent pouvoir être facilement échangés par d'autres modèles compatibles (équivalents en termes de couplage) leur permettant ainsi d'être développés parallèlement. Bien sur, à terme, l'architecture doit permettre la résolution précise et stable du couplage, à partir de schéma existant ou créé par le modélisateur. Le modélisateur créant le couplage, n'ayant pas nécessairement un profil numéricien, peut ainsi se concentrer sur sa tâche première, la modélisation physique. L'architecture permet de séparer les modèles physiques de leurs couplages à travers le schéma. L'objectif atteint, l'architecture permet de *minimiser les efforts* fournis par l'application, le modèle ou le modélisateur sur la résolution du couplage et de maximiser les possibilités offertes par l'approche partitionnée en matière de *modularité*, de *développement en parallèle* et d'*évolutivité*.

La création de modèles pour le couplage. Lors de la résolution du couplage, un certain nombre d'opérations sont faites sur les modèles, par exemple leur résolution sur un macro pas de temps, leur restauration dans un état sauvegardé (par exemple l'état en début de pas de temps), etc. Ces opérations doivent apparaître dans un contrat donné par l'architecture et que doivent remplir tous les modèles du couplage. Ce contrat doit être suffisamment exhaustif pour permettre l'utilisation de n'importe quels schémas de couplage. Certaines opérations, spécifiques à chaque modèle, doivent être implémentées par le modélisateur. Au maximum, l'implémentation logicielle de ces opérations doit être donnée par l'architecture pour minimiser le travail fourni par le modélisateur lors de la création d'un modèle et des couplages.

La création de nouveau schéma de résolution du couplage et de synchronisation. La plate-forme dans laquelle s'intègre l'architecture est une plate-forme de recherche dans laquelle apparaissent régulièrement de nouveaux problèmes de couplage. Ces couplages peuvent ne pas pouvoir être correctement résolus par les schémas existant. L'architecture doit donc faciliter la création de nouveau schéma, à partir ou non de schémas existant.

Ci-après, les objectifs de l'architecture sont détaillés et donnés en termes d'exigences. L'atteinte de ces objectifs passe par la satisfaction de ces exigences. Deux types d'exigences sont distinguées : les exigences *fonctionnelles*, ce que doit faire l'architecture, et les exigences *non-fonctionnelles*, comment elle le fait et sous quelles contraintes (imposées par le contexte AG).

3.1.2 Spécifications des exigences fonctionnelles de l'architecture

Les exigences fonctionnelles de l'architecture expriment ce que *doit faire* l'architecture et détaillent ses fonctions utiles permettant de satisfaire les objectifs précédemment donnés. Les fonctionnalités essentielles que doit supporter l'architecture sont : la communication des données d'interface des modèles, la possibilité d'avoir différentes stratégies de couplage des modèles interchangeables, un mécanisme générique d'entrées et de sorties des modèles, un ensemble d'opérations prédéfinies sur les modèles.

La communication des données d'interface. Les modèles, ou solveurs, appelés $\mathcal{M}_i$ dans le chapitre précédent 2 peuvent à priori prendre n'importe quelle forme. Pour un même phénomène physique, différents solveurs interchangeables peuvent même exister : on peut disposer pour un même modèle physique de plusieurs implémentations traduisant la richesse de la modélisation (modèle *grossier*, modèle *fin*, etc.). Néanmoins, les modèles du couplage doivent être capable de communiquer et d'échanger leurs données. Leurs données correspondent aux variables d'interface. L'architecture doit fournir à chaque solveur des moyens adaptés de communication. Ces communications vont permettre de transférer et, si nécessaire, de transformer les données (par exemple une projection sur un maillage) d'un solveur vers un autre solveur.

Les stratégies de couplage. Les modèles étant capables de communiquer, il faut maintenant pouvoir définir une stratégie de communication entre eux. Dans l'architecture, la stratégie prendra la forme d'une entité centrale de contrôle du couplage : elle permettra de définir le couplage entre modèles et de gérer sa résolution sur la macro boucle en temps. De base, l'architecture doit pouvoir fournir un ensemble de stratégies basant leur résolution sur les schémas SCE et SCI présentés dans le chapitre 2.

Un mécanisme générique d'entrées et de sorties des modèles. Des modèles ayant les mêmes interfaces doivent être aisément branchés et débranchés dans les stratégies de couplage pour être inter-changés et permettre l'évolutivité des modèles et des applications. La stratégie ne doit voir les modèles que comme des boîtes noires avec certaines entrées et certaines sorties : les variables d'interface. L'architecture doit donc définir un mécanisme générique d'entrées et de sorties que les modèles du couplage doivent supporter. Ce mécanisme doit notamment permettre de déclarer le nom, le type et la nature de l'entrée ou de la sortie.

Les opérations sur les modèles. Durant la résolution du couplage sur un macro pas de temps, les modèles constituant le couplage doivent par exemple être prédits, résolus, sauvegardés ou restaurés à un certain instant, etc. L'architecture doit définir de manière exhaustive l'ensemble de ces opérations

permettant aux stratégies de résolution de supporter n'importe quel schéma de couplage SCE ou SCI. De plus, l'architecture doit fournir un squelette de modèle supportant la plus grande partie de ces opérations. Certaines opérations, étant dépendantes du modèle considéré, seront à la charge du modélisateur lors de la création du modèle.

3.1.3 Spécifications des exigences non-fonctionnelles de l'architecture

Les exigences non-fonctionnelles caractérisent des *propriétés désirées* de l'architecture telles que sa robustesse, sa performance ou correspondent à des contraintes imposées sur l'architecture. Ce sont des exigences qui ne concernent pas spécifiquement l'architecture au contraire des exigences fonctionnelles.

Maintenabilité. L'architecture doit être maintenable à moindre coût, c'est à dire que les corrections et les transformations sur celle-ci doivent pouvoir être faites à moindre effort. Dans un cadre industriel avec des livraisons quasi-annuelles, le coût de maintenance doit être minimisé pour favoriser l'effort de *R&D*. Par ailleurs, la multiplication d'objets (des modèles physiques ou applications industrielles) dans la plate-forme ne doit pas trop impacter les coûts de maintenance de la plate-forme. La maintenabilité est notamment obtenue par l'utilisation dans l'architecture logicielle de patrons de conception, de structures de données adaptées et de suffisamment d'encapsulation.

Évolutivité. Avec les exigences de maintenabilité viennent les exigences d'évolutivité : l'architecture doit être extensible à de nouvelles fonctionnalités et ceci à moindre effort. Dans une plate-forme de recherche, cela paraît naturel. Par exemple, les stratégies de couplage de base de l'architecture peuvent ne pas être adaptées à certains problèmes et doivent pouvoir être étendues à de nouvelles stratégies. Certaines stratégies de couplage peuvent nécessitées des opérations sur les modèles autres que celles prédéfinies par l'architecture. Dans ce cas, les modèles de base doivent pouvoir être étendus pour leur ajouter ces opérations. C'est d'autant plus le cas dans le contexte AG : la *R&D* sur la phénoménologie des accidents est *très active* car elle est encore mal connue et il est donc nécessaire de faire évoluer et de rajouter des connaissances à la plate-forme (par exemple des modèles physiques) au cours de son temps de vie.

L'intégration dans la plate-forme PROCOR. L'architecture de couplage doit être intégrée de manière non-invasive dans la plate-forme PROCOR. À terme et à la fin de l'intégration, l'architecture doit apparaître dans la plate-forme comme si elle était déjà présente initialement à sa création. Son intégration doit être transparente pour le code déjà présent et la migration de ce code vers la nouvelle architecture doit pouvoir se faire progressivement pour répartir

sur le temps le développement lié à la migration. De plus, une intégration non-invasive signifie aussi la réutilisation de fonctionnalités existantes de la plate-forme pour éviter les doublons et le code inutile.

Reproduction des résultats. Dans le contexte industriel de la plate-forme, la non-régression des résultats est essentielle vis-à-vis des études déjà produites et des résultats obtenus. Le plus possible, les problèmes couplés déjà présents dans la plate-forme doivent pouvoir être recréés sous la nouvelle architecture et les résultats donnés par ces problèmes sous la nouvelle architecture doivent reproduire les résultats donnés par les problèmes initiaux.

3.2 La plate-forme PROCOR d'avant thèse

L'analyse précédente a permis de comprendre la manière dont les applications PROCOR, les modèles et les modélisateurs utilisent l'architecture. Ont ensuite pu être exprimées les différentes exigences imposées sur l'architecture. Ces besoins permettront de concevoir l'architecture et son interface avec la plate-forme PROCOR.

Cependant, comme exprimé précédemment, son intégration dans la plate-forme doit être *non-invasive* : l'architecture doit réutilisée des fonctionnalités de la plate-forme pour éviter, entre autres, les doublons dans le code. Par conséquent, la conception de l'architecture doit être faite avec une compréhension au moins partielle de l'architecture de la plate-forme PROCOR originelle. En particulier, les exigences fonctionnelles et non-fonctionnelles soulèvent plusieurs questions concernant la plate-forme :

— Quels sont les *types* de modèles existant dans la plate-forme ?
— Quels sont les mécanismes et fonctionnalités sur les *entrées* et les *sorties* des modèles et les *types* de ceux-ci déjà présents dans la plate-forme ?
— Quels sont les *opérations* déjà supportées par les modèles de la plate-forme ?

Pour répondre à ces questions, la plate-forme PROCOR *originelle* est présentée ci-après.

3.2.1 Une plate-forme de calculs physiques et statistiques

La plate-forme logicielle CEA PROCOR [69] est structurée en deux parties : une partie logicielle statistique et une partie logicielle physique.

La partie logicielle statistique permet des calculs Monte-Carlo sur les applications PROCOR et l'analyse des résultats par un ensemble d'outils d'analyse de sensibilité et d'incertitude. En particulier, les calculs Monte-Carlo sont délégués à la plate-forme de calculs d'incertitudes et de sensibilité CEA URANIE [43]. Un exécutable C++ permet le couplage de la plate-forme URANIE

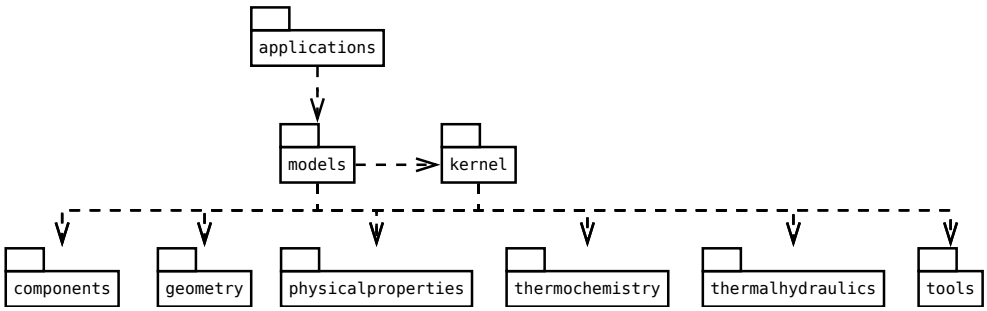


FIGURE 3.3 – Packages de la plate-forme PROCOR.

avec les entrées des applications PROCOR et un échantillonnage sur ces entrées. Les outils d'analyse de sensibilité et d'incertitude correspondent à un ensemble de scripts CINT, interpréteur C/C++ pour la plate-forme ROOT d'analyse de données développée au CERN [3] sur laquelle se base la plate-forme URANIE. Cette partie n'a pas fait l'objet de modifications durant le thèse.

La partie logicielle physique est développée en `Java` utilisant le paradigme de programmation orientée objet. L'ensemble des classes de la partie physique sont testées unitairement par le programme JUnit [58]. Les classes qui ne peuvent être testées unitairement font généralement l'objet de tests de non-régression. Par l'utilisation des bibliothèques JNI ou JNA, offrant des possibilités d'interfaçage avec des bibliothèques partagées, la plate-forme physique PROCOR est capable d'utiliser, entre autres, des codes tels que TOLBIAC-ICB [90] pour le calcul de propriétés physiques ou OpenCalphad [92] pour le calcul de propriétés thermodynamiques.

La partie physique de la plate-forme PROCOR est celle dans laquelle va s'intégrer l'architecture de couplage. De ce fait, par la suite, "plate-forme PROCOR" désignera uniquement cette partie de la plate-forme. Une description de son architecture logicielle abordée sous le prisme du couplage et de la synchronisation est donnée ci-après. Y sont décrits les objets de la plate-forme susceptibles d'interagir avec la nouvelle architecture logicielle et apportant des éléments de réponses aux questions posées en début de section.

3.2.2 Vue d'ensemble des packages de PROCOR

La plate-forme est divisée en un ensemble de packages `Java` décrits dans le diagramme UML de la figure 3.3. Les principaux packages sont les packages `applications`, `kernel`, `models`, `external`, `components`, `physicalproperties`, `geometry`, `thermalhydraulics`, `thermochemistry`, `tools`. Dans ce diagramme, les dépendances entre packages sont représentées par une flèche en pointillée dirigée du package dépendant au package fournissant un service. Seules les

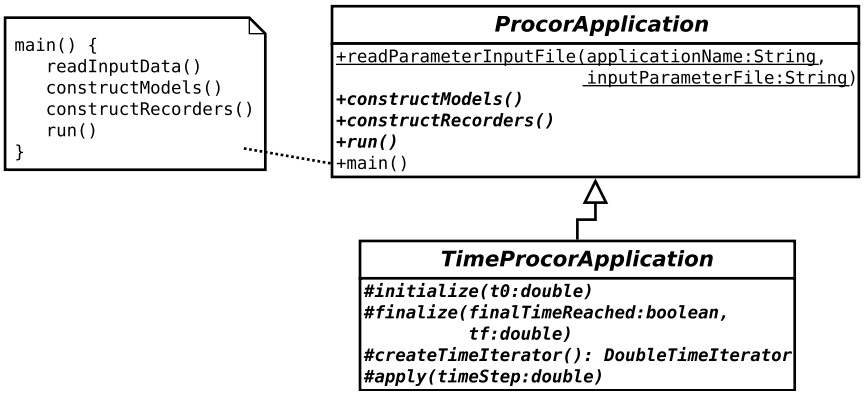


FIGURE 3.4 – Classes PROCOR relatives aux applications.

dépendances principales sont représentées. On notera la topologie verticale du diagramme et l'absence de cycle de dépendances entre packages, soulignant une architecture facilement maintenable et bien construite.

Les packages les plus bas dans la diagramme contiennent les classes élémentaires et les fonctionnalités fondamentales de la plate-forme (par exemple pour la géométrie des éléments d'un modèle, les propriétés physiques des matériaux d'un modèle ou encore les lois de fermeture d'un modèle données par des corrélations de thermohydraulique (voir l'annexe A pour des exemples de lois de fermeture)).

Plus haut, le package `kernel` contient les classes de base pour les modèles physiques et la gestion de leurs entrées.

Le package `models` contient les modèles physiques de la plate-forme PROCOR utilisant les classes de modèles du package `kernel`.

Enfin, le package `applications` contient l'ensemble des applications industrielles PROCOR construites à partir des modèles physiques.

Les applications PROCOR, du package `applications`, sont celles définissant les systèmes complexes constitués des modèles couplés. Le package `kernel` contient les classes de base de modèles ainsi que les classes gérant leurs entrées. Ces quelques classes sont par conséquent intéressantes pour l'architecture de couplage, leur fonctionnement est détaillé ci-après.

3.2.3 Les classes d'applications

Le package `applications` contient les classes abstraites `ProcorApplication` et `TimeProcorApplication` (applications dépendantes du temps) dont chaque application de la plate-forme hérite. Ces classes sont décrites par le diagramme UML de la figure 3.4.

Une application PROCOR est résolue du temps initial au temps final en appelant sa méthode `main`. Son pseudo-code est donné dans le diagramme de la figure 3.4. Dans cette méthode, l'application commence par lire ses paramètres d'entrées données par l'extérieur (`readInputData`), puis construit les modèles physiques la constituant (`constructModels`), puis construit les objets permettant d'enregistrer ses résultats (`constructRecorders`) et est enfin résolue (`run`).

L'implémentation logicielle dans les `TimeProcorApplication` de la méthode `run` est donnée par l'algorithme 6. Un itérateur en temps, permettant

Algorithme 6 Implémentation par les `TimeProcorApplication` de la méthode `run`.

```
1: iterator ← createTimeIterator()
2: t0 ← iterator.init()
3: initialize(t0)
4: while iterator.hasNext() do
5:     timeStep ← iterator.next()
6:     apply(timeStep)
7: end while
8: finalTimeReached ← iterator.isFinalTimeReached()
9: tf ← iterator.getCurrentTime()
10: finalize(finalTimeReached)
```

par exemple de fixer des points de rendez-vous temporels, boucle de l'instant initial, auquel l'application est initialisée, jusqu'à l'instant final, auquel des opérations sont faites (par exemple l'écriture dans les fichiers de sorties). À chaque macro pas de temps (dans la boucle ligne 6 de l'algorithme), l'application est avancée dans le temps en *résolvant le couplage entre ses modèles*.

En pratique, chaque application PROCOR dépendante du temps implémente la méthode `constructModels` dans laquelle elle déclare les modèles couplés la constituant puis implémente la méthode `apply` dans laquelle sont *figés* l'ordre de résolution des modèles sur un macro pas de temps et les communications entre ceux-ci. Dans la plate-forme originelle, les modèles sont chaînés et les communications écrites de manière *ad-hoc* pour chaque couplage.

3.2.4 Les classes de modèles et de leurs paramètres

Le package `kernel` contient les classes de base de PROCOR pour la construction des modèles et la gestion de leurs paramètres. En particulier, ce package contient :

— des classes relatives à la gestion des paramètres.
— des classes relatives aux modèles décrits par un ensemble d'équations différentielles ordinaires (EDO) ou décrits par un ensemble d'équa-

tions différentielles algébriques (EDA, une équations différentielle algébrique étant formée d'une EDO et d'une ou plusieurs équations algébriques (EA)).
— des classes relatives aux modèles à paramètres ne rentrant pas sous le paradigme des équations différentielles.

La gestion des paramètres. Un mécanisme générique permettant aux objets de la plate-forme de définir un ensemble de paramètres dont les valeurs, données à l'extérieur de l'objet, peuvent être utilisées à l'intérieur de l'objet. Ce mécanisme est par exemple utilisé par les applications PROCOR lisant depuis un fichier la valeur de leurs paramètres (voir pour `readInputData` dans section précédente). En particulier, ils rendent possibles des calculs de type Monte-Carlo avec la partie statistiques de PROCOR échantillonnant les paramètres d'entrées des applications.

Les modèles `ODEModel`. Ces modèles peuvent être décrits par un ensemble d'équations différentielles ordinaires (EDO) du premier ordre et d'équations algébriques (EA). Ces équations utilisent le précédent mécanisme de gestion des paramètres et un mécanisme spécifique de gestion de valeurs pour le calcul de leurs variables. Les équations des modèles `ODEModel` prennent la forme suivante

$$\begin{cases} \dfrac{\mathrm{d}y}{\mathrm{d}t} = \mathcal{F}(y, x, t) \\ \mathcal{G}(y, x, t) = 0 \end{cases} \tag{3.1}$$

dans laquelle y représente les variables du système, x les paramètres du système et t le temps. La fonction $\mathcal{F}$ permet de définir les EDO du système tandis que la fonction $\mathcal{G}$ représente les EA du système. La combinaison des EDO et EA forment les équations différentielles algébriques (EDA). Ces modèles utilisent la bibliothèque Apache Commons Math [4], notamment pour l'intégration en temps des équations, pour la description des modèles à états et pour gérer les transitions en interne des modèles entre les états du modèle avec des fonctions de seuil notées g (à la suite d'une transition, le modèle mute et change de système EDA avec de nouvelles fonctions $\mathcal{F}$ et $\mathcal{G}$).

Les modèles à paramètres (classe `ParameterizedModel`). Ces modèles généraux ne peuvent être décrits par des EDO et peuvent prendre n'importe quelle forme, par exemple modèles stationnaires, modèles réduits (méta modèle, réseaux de neurones, etc.), modèles appelant un code externe à la plate-forme PROCOR, etc. Ils bénéficient néanmoins du mécanisme de gestion des paramètres pour leurs paramètres d'entrées.

Le mécanisme de gestion des paramètres d'entrées des modèles est générique et peut être réutilisé pour l'architecture de couplage. Donnée ci-après,

la description des `ODEModel`, ayant un besoin intensif en paramètres, permet d'identifier les fonctionnalités essentielles que doit supporter le mécanisme de paramètres. Une fois ces fonctionnalités identifiées, l'architecture logicielle relative aux classes de paramètre est détaillée.

3.2.4.1 Les classes de paramètres au travers des modèles `ODEModel`

La classe abstraite `ODEModel` de la plate-forme `PROCOR` représente les modèles `ODEModel`. Ces modèles sont composés

— d'un ensemble d'états.
— de transitions entre états.

Les états des modèles `ODEModel` déclenchent des événements lorsque qu'une fonction de seuil, notée g, passe par zéro (par exemple lorsque le modèle disparaît et que sa masse devient nulle). La transition amène le modèle dans un nouvel état avec de nouvelles équations généralement différentes des équations de l'état précédent.

Chaque état du modèle a un ensemble d'EDO. Ces EDO sont généralement des équations de conservation de la masse, de l'énergie ou de l'enthalpie obtenues par intégration des équations aux dérivées partielles sur l'ensemble du domaine considéré (menant à des modèles 0D ou "lumper parameter", voir [70] pour un exemple de tels modèles). Des équations algébriques (EA) viennent ensuite fermer le système d'EDO. L'idée étant de créer une bibliothèque d'équations algébriques qui peuvent être réutilisées plusieurs fois dans différents modèles. L'ensemble de ces équations (les EDO et EA) forment alors un ensemble d'équations différentielles algébriques (EDA) (voir [82] pour des informations sur les EDO et EDA).

De manière générale, une équation dans `PROCOR` a des variables, notées y, calculées à partir de paramètres, notés x, à un instant t.

Les EDO sont représentées par une fonction $\mathcal{F}$ et peuvent être écrites sous la forme

$$\frac{\mathrm{d}y}{\mathrm{d}t} = \mathcal{F}(t, x, y).$$

Les EA sont représentées par une fonction $\mathcal{G}$ et peuvent être écrites sous la forme

$$\mathcal{G}(t, x, y) = 0.$$

La figure 3.5 donne un exemple d'`ODEModel` de la plate-forme `PROCOR`. Il est composé de trois états. Chaque état a des EDO et des EA *différentes*. Par exemple, l'état 1 a deux EDO de conservation de la masse et de l'énergie et plusieurs EA calculant les fermetures de ces premières équations (voir l'annexe A pour des exemples de lois de fermeture). Les variables y sont ici la masse $\mathbf{m}$ et la température $\mathbf{T}$. Les autres quantités sont les paramètres x, sauf

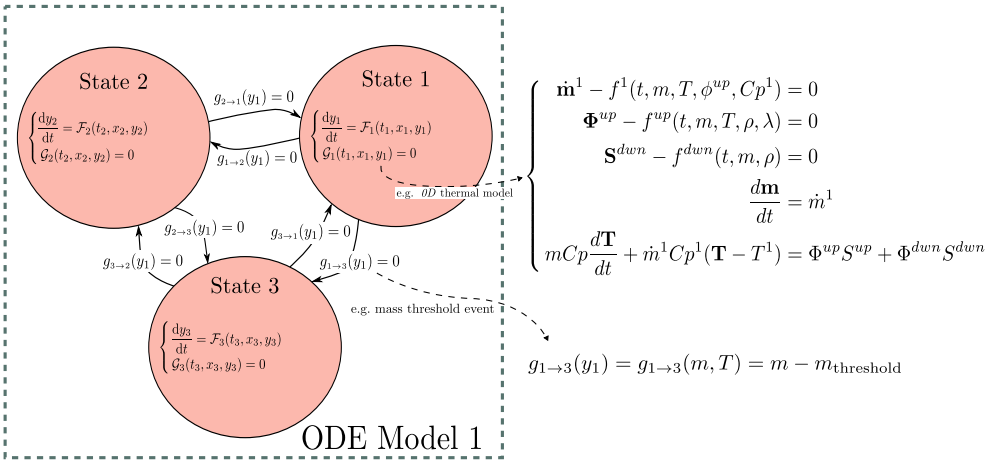


FIGURE 3.5 – Représentation schématique d'un modèle implémentant la classe `ODEModel` de `PROCOR`, de ses états, des équations des états (EDO et EA) et des transitions entre états (flèches en pointillés avec les fonctions de transition).

le temps noté t. Les transitions entre états sont gérés par les fonctions de seuil $g_{?\to?}$ lors de leur passage par 0.

Pour permettre l'implémentation *logicielle* de cet exemple dans la plate-forme, les paramètres doivent pouvoir supporter trois fonctionnalités :

— Les deux équations différentielles et trois équations algébriques utilisent des paramètres représentant la même propriété physique, par exemple la densité ρ, la capacité calorifique Cp, etc. Ces paramètres doivent pouvoir être réduits à la même valeur au cours du calcul.

— Un paramètre d'une équation peut être calculé par une autre équation. Par exemple, l'EA

$$\Phi^{up} - f^{up}(t, m, T, \rho, \lambda) = 0$$

calcule le paramètre Φ^{up} de l'EDO de conservation d'énergie. Une association doit pouvoir être possible entre la valeur calculée par une équation et le paramètre d'une autre.

— La valeur d'un paramètre peut aussi être donnée depuis l'extérieur, par exemple pour les paramètres Φ^{dwn}, T^1, Cp^1, S^{up} de l'EDO de conservation de d'énergie

$$mCp\frac{d\mathbf{T}}{dt} + \dot{m}^1 Cp^1(\mathbf{T} - T^1) = \Phi^{up}S^{up} + \Phi^{dwn}S^{dwn}$$

ou ρ pour l'EA précédente calculant Φ^{up}.

En résume, le mécanisme des paramètres dont se sert les `ODEModel` doit offrir trois fonctionnalités essentielles :

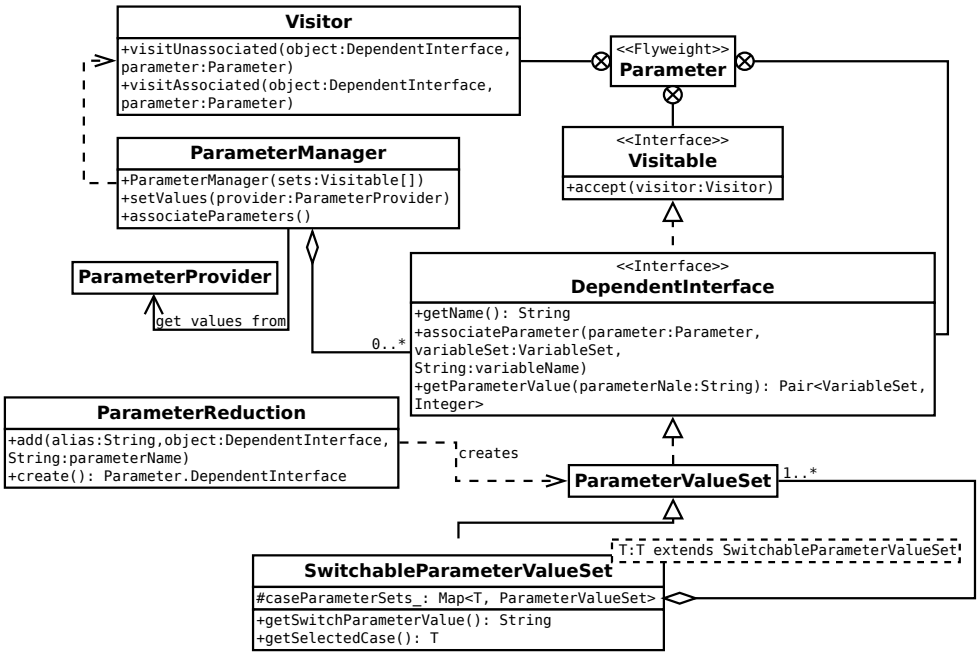


FIGURE 3.6 – Architecture du mécanisme de gestion des paramètres dans la plate-forme PROCOR.

— Plusieurs paramètres doivent pouvoir être *réduits* en un seul paramètre car ils représentent la même quantité physique .
— Un paramètre doit pouvoir être *associé* à une variable calculée par une autre équation de l'état.
— Un paramètre doit pouvoir être *associé* à une variable imposée depuis l'extérieur du modèle.

3.2.4.2 L'architecture logicielle des classes de paramètres

Pour supporter les précédents mécanismes, un ensemble de classes génériques articulées autour de la gestion des paramètres existent dans la plate-forme PROCOR. Elles sont décrites dans le diagramme UML de la figure 3.6. Les classes centrales du diagramme sont les classes `Parameter`, `Parameter.DependentInterface` et `ParameterValueSet`.

La classe `Parameter`. Elle est utilisée par les modèles pour décrire leurs paramètres, par un nom, un type et une dimension. Pour optimiser l'usage mémoire lors de la création d'un grand nombre de paramètres, un patron de conception *poids-mouche* [38] est utilisé pour la classe `Parameter`.

La classe `Parameter.DependentInterface`. Une classe qui déclare des paramètres implémente la classe abstraite `DependentInterface`. Une classe qui

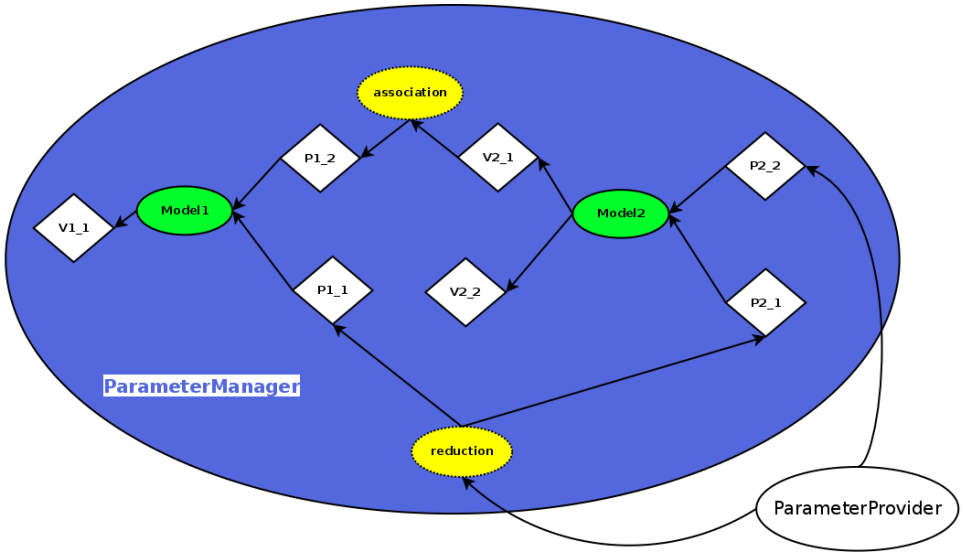


FIGURE 3.7 – Exemple d'utilisation des mécanismes d'*association* et de *réduction* de paramètres.

ne fait qu'utiliser des paramètres implémente la classe `Visitable`. Par exemple les équations des états des modèles `ODEModel` déclarent des paramètres et implémentent donc `DependentInterface`. Les états gérant ces équations implémentent `Visitable` de telle sorte que ses équations peuvent lui déléguer la gestion de leurs paramètres. Un objet voulant interagir avec un objet `Visitable` utilisant des paramètres doit implémenter l'interface `Visitor`. Un patron de conception *Visiteur* [38] est utilisé pour la classe `Parameter`.

La classe `ParameterValueSet`. Des objets de cette classe, réalisant l'interface `DependentInterface`, déclarent des paramètres et peuvent leur assigner des valeurs. D'autres classes déclarant des paramètres peuvent déléguer, par association, la gestion de leurs paramètres à de tels objets. Un patron de conception *proxy* [38] est utilisé pour cette classe. La sous classe `SwitchableParameterValueSet` hérite de `ParameterValueSet`. Elle a un paramètre spécial agissant comme un *switch*. À l'activation de ce *switch*, de nouveaux paramètres sont ajoutés à la classe. Un patron de conception *état* [38] est utilisé. En pratique, cette classe est utilisée pour la gestion des versions des modèles.

Ces classes permettent les fonctionnalités d'*association* et de *réduction* de paramètres décrites précédemment pour les modèles `ODEModel`.

Un exemple de leurs utilisations est donné dans la figure 3.7. Dans cet exemple, deux objets `Model1` et `Model2` réalisant l'interface `DependentInterface` sont liés. Chaque modèle a deux paramètres (respectivement `P1_1`, `P1_2` et `P2_1`, `P2_2`) et calcul des valeurs (`V1_1` pour `Model1` et `V2_1`, `V2_2` pour `Model2`).

De plus,

— les paramètres `P1_1` et `P2_1` représentent la même quantité physique, ils ont été réduits.

— le paramètre `P1_2` est donné par le calcul de `V2_1`, ces deux objets ont été associés.

Un objet `ParameterProvider` permet de donner une valeur au paramètre `P2_2` et aux paramètres réduits `P1_1` et `P2_1`. Ainsi, tous les paramètres des deux objets `Model1` et `Model2` ont une valeur.

L'implémentation logicielle de ces fonctionnalités dans la plate-forme est décrite ci-après.

L'association avec les paramètres. Il est possible d'assigner une valeur au paramètre d'un objet `DependantInterface` en faisant appel à sa méthode `associateParameter` avec un objet `VariableSet` (décrit dans le diagramme UML de la figure 3.8). Se faisant, une association entre la valeur stockée dans cet objet et le paramètre du modèle est faite de telle sorte que toutes les modifications faites sur la valeur seront vues par le modèle. Les équations des états des modèles `ODEModel` sont des objets `VariableSet` et des `DependentInterface`. Ainsi, le paramètre d'une équation peut être calculé par une autre équation en l'*associant* à la valeur calculée par l'équation. En pratique, les paramètres sont soit associés à des objets `VariableSet` correspondant à des équations, dans ce cas la valeur des paramètres est calculée automatiquement par l'équation et stockée dans le `VariableSet`, ou à des objets `VariableSet` créés par l'utilisateur, dans ce cas la valeur des paramètres est fixée manuellement par l'utilisateur. Dans le second cas, un objet `ParameterManager` permet de gérer les paramètres encore non associés et de leur donner des valeurs par le biais d'objets `ParameterReduction` créés par l'utilisateur.

La réduction de paramètres. La classe `ParameterReduction` est utilisée lorsque des paramètres déclarés par différents objets représentent la même quantité physique et doivent donc être réduits à la même valeur au cours du calcul. Dans ce cas, un objet `ParameterReduction` est créé et sa méthode `add` est utilisée pour créer un lien entre les paramètres et un nouveau paramètre unique attaché à l'objet `DependantInterface` créé par la méthode `create`. Par la suite, une valeur donnée à ce paramètre unique sera redistribuée vers les autres paramètres des objets.

Les mécanismes d'association et de réduction utilisent abondamment les objets `VariableSet`. Les classes relatives à ces objets sont décrites dans le diagramme UML de la figure 3.8. Les objets `VariableSet` sont utilisés pour sauvegarder des valeurs par leur nom, leur type et leur dimension. Ces objets sont enregistrés et ensuite gérés par un `VariableSetManager`. Un patron de conception *Publisher/Subscriber* [38] est utilisé. Les valeurs des `VariableSet`

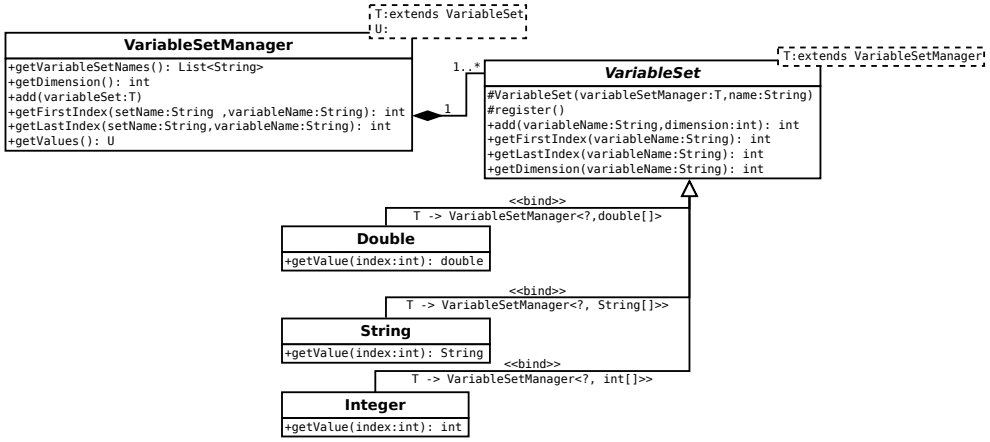


FIGURE 3.8 – Architecture des classes relatives au stockage de valeurs dans des `VariableSet` dans la plate-forme PROCOR.

sont toutes gérées par un et unique objet, le manager, qui peut choisir librement la manière de les stocker. Puisque `Java` ne gère pas correctement les tableaux de tableaux, voir [48] [79] par exemple, le manager peut choisir de sauvegarder les valeurs dans un tableau à une dimension et d'utiliser des redirections. En particulier, la stratégie utilisée dans la classe `VariableSetManager` est celle conseillée dans [20].

La valeur des paramètres est stockée par des objets `VariableSet`, leur type `Java` est donc limité à ceux gérés par les classes `VariableSet`. Dans PROCOR, le type des paramètres se limite au tableau de doubles `double[]`, au tableau d'entiers `int[]` ou au tableau de chaînes de caractères `String[]`. Les objets `Java` *complexes* ne peuvent pas être des paramètres.

3.2.4.3 Retour sur l'architecture logicielle des modèles `ODEModel`

L'utilisation des classes précédentes de gestion des paramètres est illustrée au travers des modèles `ODEModel`. Les classes PROCOR associées à ces modèles sont décrites par le diagramme UML de la figure 3.9. Un modèle `ODEModel` peut avoir plusieurs états représentés par des objets de la classe `ODEModelState`. Ces états ont un système d'équations différentielles ordinaires (l'attribut `firstOrderEqSet_` et un système d'équations algébriques (l'attribut `functionEqSet_`). Ces systèmes d'équations sont représentés par des objets de la classe `EquationSystem` eux-mêmes composés de plusieurs équations représentées par des objets de la classe abstraite `Equation`. Les EDO sont représentées par des objets de la classe `FirstOrderEquation` et les EA sont représentées par des objets de la classe `FunctionEquation`.

La classe `Equation` implémente l'interface `Parameter.DependentInterface`, car une équation déclare des paramètres, et hérite de la classe `VariableSet`

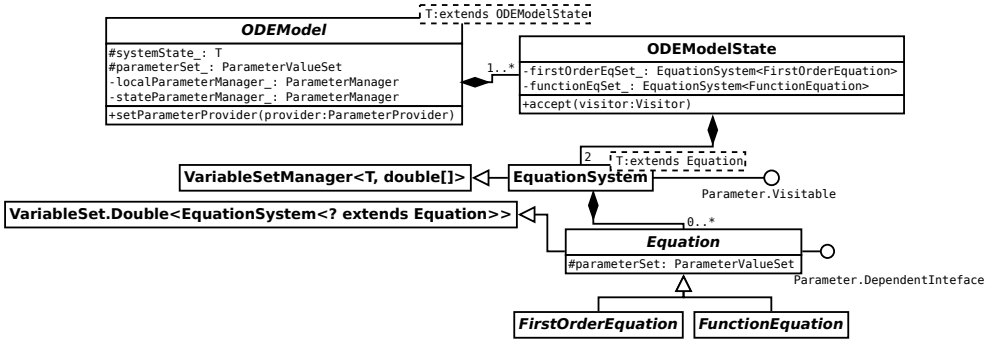


FIGURE 3.9 – Architecture des classes relatives à l'utilisation du mécanisme de gestion des paramètres dans les modèles `ODEModel` de PROCOR.

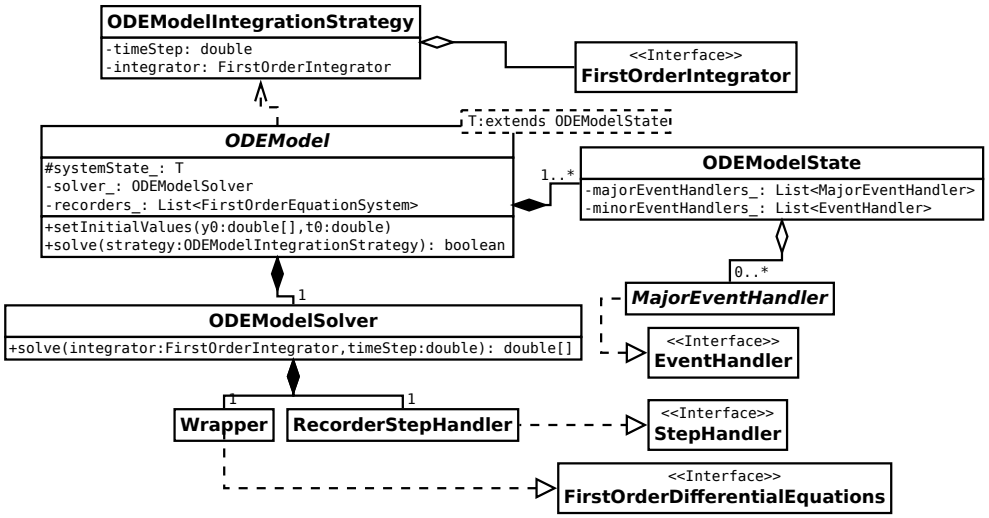


FIGURE 3.10 – Architecture des classes relatives à la résolution des modèles `ODEModel` de PROCOR.

car elle calcule des valeurs. En héritant de ces classes, les équations des états des modèles `ODEModel` peuvent utiliser le mécanisme d'association et de réduction des paramètres :

- Plusieurs paramètres des équations peuvent être *réduits* en un seul paramètre car ils représentent la même quantité physique .
- Un paramètre d'une équation peut être *associé* à une variable calculée par une autre équation de l'état.
- Un paramètre d'une équation peut être *associé* à une variable imposée depuis l'extérieur du modèle.

Les classes relatives à la résolution des modèles `ODEModel` sont données par le diagramme UML de la figure 3.10. Un ensemble de classes ont été développées spécifiquement pour faire *interface* avec la bibliothèque Apache Com-

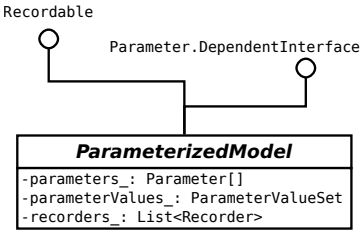


FIGURE 3.11 – Classe représentant les modèles à paramètres de PROCOR.

mon Maths [4]. En particulier les classes PROCOR `Wrapper` et `RecorderStepHandler` pour l'intégration en temps et l'enregistrement des résultats des modèles réalisent les interfaces `StepHandler` et `FirstOrderDifferentialEquations` de la bibliothèque, ou encore la classe PROCOR `MajorEventHandler` pour la gestion des événements et transitions entre états réalisent l'interface `EventHandler` de la bibliothèque.

Pour la résolution en temps des modèles `ODEModel`, une patron de conception *stratégie* [38] est utilisé. L'utilisateur contrôle la résolution avec un objet `ODEModelIntegrationStrategy` donné en paramètre de la méthode `solve`. En plus du macro pas de temps de résolution, un tel objet contient l'intégrateur en temps réalisant l'interface `FirstOrderIntegrator` de la bibliothèque Apache Commons Math. En interne, un modèle `ODEModel` délègue sa résolution à un objet `ODEModelSolver` qui utilisent les classes réalisant les interfaces de la bibliothèque Apache Commons Maths. Des objets `ODEModelRecorder` attachés à un modèle `ODEModel` permettent la sauvegarde des valeurs des équations à chaque micro pas de temps d'intégration, par exemple dans un fichier au format `csv`.

3.2.4.4 Conclusion sur les modèles et leurs paramètres

Le mécanisme de gestion des paramètres de PROCOR a été présenté au travers des classes de modèles `ODEModel`. Ce mécanisme est utilisé par bien d'autres classes de la plate-forme, par exemple les applications PROCOR ou les modèles à paramètres. Ces derniers sont représentés dans la plate-forme par les classes `ParameterizedModel` décrites figure 3.11. Ils permettent aux modèles qui ne peuvent être décrits par des équations différentielles algébriques comme les modèles `ODEModel` d'utiliser des paramètres et d'être enregistrés par des objets `Recorders`. Généralement, ces modèles utilisent en interne d'autres sous-modèles qui peuvent récursivement être des modèles à paramètres ou des modèles `ODEModel`. L'ensemble des modèles forment un arbre de modèles (les modèles `ODEModel` n'ayant pas de sous-modèle ne peuvent être que des feuilles de l'arbre). Les modèles à paramètres, pour être résolu sur un macro pas temps, doivent donc résoudre un problème de couplage. En pratique, la résolution est faite dans une méthode fixée par le modélisa-

teur dans laquelle les sous-modèles sont couplés en temps et communiquent.

Le mécanisme de paramètre offre de nombreuses possibilités en terme de communication entre modèles et permet déjà un large éventail de fonctionnalités. Néanmoins, il ne permet de communiquer que des tableaux de types `Java` *simples* `double[]`, `int[]` ou `String[]`, et pas d'objets `Java` plus complexes, ce qui peut être *limitant* pour la communication entre modèles.

3.2.5 Conclusion et intégration de l'architecture dans la plate-forme

Suite à la présentation de la plate-forme d'avant thèse, on peut faire le constat suivant :

— le mécanisme des paramètres offre un moyen de communiquer *seulement certains types* de valeurs entre modèles.

— les applications `PROCOR` ainsi que les modèles à paramètres représentent *un système complexe* qui n'est pas formellement représenté dans la plate-forme mais qui est fixé manuellement en interne. L'ensemble des sous-modèles forme un arbre de modèles couplés (les modèles `ODEModel` étant généralement cachés et utilisés en interne de modèles à paramètres et sont des feuilles de l'arbre). D'une part, le couplage entre modèles et l'ordre dans lequel sont résolus l'ensemble des modèles sont *fixés* en interne des modèles (les modèles sont en général chaînés). D'autre part, l'arbre des modèles n'a pas de représentation logicielle mais est déclaré de manière implicite par chaque modèle. Il n'est pas visible de l'extérieur des objets et aucune opération n'est possible dessus (par exemple la modification d'un modèle de l'arbre par un autre).

En résumé, dans les classes de la plate-forme *originelle*, aucune ne permet une représentation satisfaisante des systèmes complexes et une résolution précise (par l'utilisation de schéma de couplage adapté) et flexible (par l'utilisation de différents schéma de couplage) des problèmes couplés sous-jacents.

Le souhait est d'introduire dans la plate-forme une nouvelle classe de modèles permettant une meilleur représentation et une meilleur résolution des systèmes complexes par l'utilisation du formalisme et des schémas numériques de couplage de l'approche partitionnée introduits dans le chapitre 2. Ces nouvelles classes doivent utiliser le plus de fonctionnalités déjà existantes dans la plate-forme (par exemple le mécanisme de gestion des paramètres).

Cette classe de modèles partitionnés sera appelée `PartitionedModel`. Pour permettre son intégration non-invasive, naturelle et progressive dans l'architecture `PROCOR`, sa position relative par rapport aux classes `ProcorApplication` et `ParameterizedModel` doit être correctement déterminée. Par position relative par rapport à une classe, il est entendu relation qu'elle entretient avec cette classe. Les deux paragraphes suivant donnent les relations entre la classe

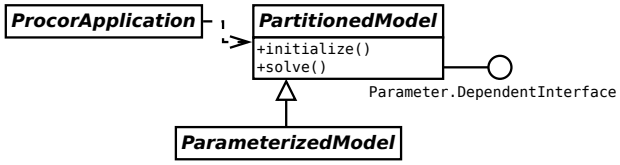


FIGURE 3.12 – Position de la nouvelle classe de modèles partitionnés dans l'architecture par rapport aux classes des applications et des modèles à paramètres de la plate-forme PROCOR.

`PartitionedModel` avec les autres classes.

***Association* avec la classe `ProcorApplication`.** À chaque application PROCOR peut être *associée* un modèle `PartitionedModel`. Lors de l'appel de la méthode `constructModels` de la méthode `main` d'une application PROCOR, donnée à la figure 3.4, le modèle partitionné est construit à partir des modèles de l'application et de leur couplage. Ensuite, lors de l'appel de la méthode `run`, le modèle partitionné est résolu de l'instant initial à l'instant final par des itérations sur la macro boucle en temps. Par exemple dans la méthode `run` des `TimeProcorApplication` donnée par l'algorithme 6, le modèle partitionné est initialisé, puis itère jusqu'à la fin de la simulation en calculant le prochain macro pas de temps de résolution, qui peut changer suivant les événements internes des sous-modèles, et en résolvant le couplage sur ce pas de temps. Une relation d'*association* existe entre la classe `ProcorApplication` et la classe `PartitionedModel`.

***Héritage* avec la classe `ParameterizedModel`.** Chaque modèle à paramètres représentent un modèle partitionné monolithique n'ayant qu'une partition : la résolution du couplage de ses sous-modèles sur un macro pas de temps est fixé par le modélisateur dans le corps d'une méthode de la classe.

Ces relations sont décrites par le diagramme UML de la figure 3.12. Pour les applications, créer et leur associer un modèle partitionné se fera aisément et la migration des applications vers la nouvelle architecture peut se faire sans difficulté. Les modèles à paramètres (classe `ParameterizedModel`) existant de la plate-forme rentreront immédiatement dans la nouvelle architecture. De plus, la relation d'héritage permet de migrer au fur et à mesure les modèles à paramètres vers de vrais modèles partitionnés de la nouvelle architecture. En pratique, pour se faire, ces modèles devront déclarer au fur et à mesure leurs sous-modèles permettant un accès à l'arbre de modèles depuis l'extérieur du modèle.

3.3 L'architecture logicielle et algorithmes

Dans les sections précédentes, l'utilisation, les besoins et les exigences de l'architecture de couplage ont été identifiés. Ont été présentés d'une part les fonctionnalités essentielles que doit supporter l'architecture (la communication des données d'interface des modèles, le choix de différentes stratégies de couplage des modèles, un mécanisme générique d'entrées et de sorties des modèles, un ensemble d'opérations prédéfinies sur les modèles) et d'autre part les exigences non-fonctionnelles et contraintes imposées sur celle-ci. Suivant ceci, une description de l'architecture de la plate-forme PROCOR d'avant thèse a permis d'identifier les mécanismes et classes déjà présents dans la plate-forme qu'il est possible de réutiliser et de comprendre le positionnement de l'architecture dans la plate-forme (les relations entre les nouvelles classes de l'architecture et les anciennes classes de la plate-forme PROCOR originelle). Le but de cette section est de détailler l'implémentation logicielle des fonctionnalités de la nouvelle architecture au sein de l'architecture originelle de PROCOR et les algorithmes implémentés dans celles-ci.

3.3.1 Les classes des modèles et de leurs entrées et sorties

Les modèles constituant un problème de couplage sont les modèles partitionnés (classe `PartitionedModel`). L'analyse précédente permet d'identifier les fonctionnalités essentielles que doivent supporter les modèles partitionnés :

- ils doivent pouvoir être manipulés lors de la résolution du couplage et doivent donc supporter un certain nombre d'opérations (notamment leur résolution faisant appel à leur solveur local et éventuellement un micro pas de temps δt).
- ils doivent pouvoir échanger et recevoir des données extérieurs (variables d'entrées et de sorties ainsi que paramètres d'entrées). Les données échangées peuvent être des objets complexes.
- ils peuvent récursivement être constitués de sous-modèles partitionnés. S'ensuit un arbre de modèles partitionnés : un modèle dispose d'une forêt de sous-modèles couplés.

Les opérations sur les modèles partitionnés. Dans [35] sont décrits des schémas de couplage. Lors de la résolution sur un macro pas de temps, un tel schéma de couplage doit manipuler les modèles couplés. Durant la résolution sur un macro pas de temps Δt, différentes techniques, décrites dans la figure 3.13 (extraite et adaptée de [35]), sont utilisées par le schéma de couplage :

- La prédiction consiste à prédire les valeurs à $t^{n+1} = t^n + \Delta t$ d'un modèle qui n'a pas encore été résolu et qui est toujours à l'instant t^n.

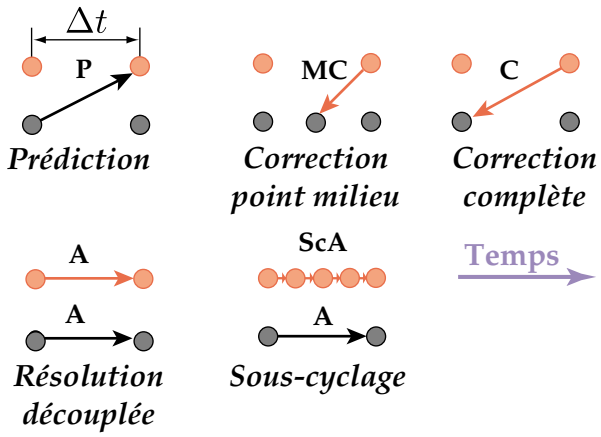


FIGURE 3.13 – Techniques utilisées par les schémas de couplage sur les modèles partitionnés en rouge et noir, figure extraite et adaptée de [35].

— Le correction consiste à restaurer à l'instant t^n un modèle résolu à l'instant t^{n+1} puis le re-résoudre avec des valeurs corrigées (la stratégie de correction point-milieu consiste à corriger la nouvelle valeur calculée en prenant en compte les anciennes valeurs calculées, la stratégie complète consiste à ne prendre que la nouvelle valeur calculée).
— La résolution découplée consiste à ne pas faire communiquer les modèles couplés durant la résolution.
— Le sous-cyclage consiste à résoudre un modèle entre t^n et $t^{n+1} = t^n + \Delta t$ par plusieurs sous-cycles de plus petites longueurs que Δt.

Pour permettre ces techniques, les modèles partitionnés doivent supporter certaines opérations. Toutes ces techniques nécessites au moins que les modèles puissent être avancés en temps avec un certain pas de temps, plus ou moins grand suivant la taille des sous-cycles. Les techniques de correction ou prédiction nécessitent la sauvegarde et la restauration à un certain instant au début, à l'intérieur ou à la fin du macro pas de temps. Pour ce faire, un modèle partitionné devra déclarer un ensemble de *variable* représentant son état. Ses variables devront pouvoir être sauvegardées puis restaurées permettant ainsi la sauvegarde et la restauration du modèle.

L'échange et la réception de données. Le mécanisme de paramètre de la plate-forme PROCOR, détaillé à la section 3.2, peut être réutilisé pour permettre à un modèle partitionné de recevoir des données. Néanmoins, ces données devront se limiter à des tableaux de types primitifs en Java `double[]`, `int[]`, ou `String[]`. Il est souhaitable que les modèles puissent échanger des objets plus complexes. Un modèle partitionné devra donc, en plus de ses variables d'état, déclarer un ensemble de paramètres et de *variables* d'interfaces.

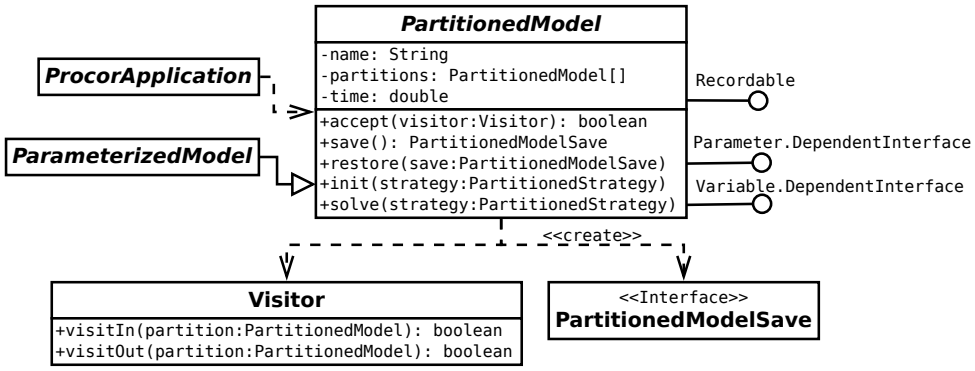


FIGURE 3.14 – Classe représentant les modèles partitionnés dans l'architecture de couplage.

Les sous-modèles des modèles partitionnés. Chaque modèle partitionné peut être constitué de sous-modèles, eux-mêmes partitionnés. Un modèle partitionné doit déclarer ses sous-modèles partitionnés pour rendre accessible depuis l'extérieur du modèle tout l'arbre de modèles résultant. Un élément de l'arbre pourra ainsi être facilement échangé ou manipulé.

3.3.1.1 Les modèles partitionnés

La classe `PartitionedModel` représente un modèle partitionné dans l'architecture de couplage. Elle est décrite dans la figure 3.14.

L'arbre des modèles partitionnés est implicitement défini et rendu publique au travers de la méthode `getPartitions`. Il peut être visité par un objet implémentant la classe `Visitor` en appelant sa méthode `accept` donnée par l'algorithme 7. Un patron de conception *Visiteur* est utilisé [38].

Algorithme 7 `model.accept(visitor)` d'un modèle partitionné.

```
1: if visitor.visitIn(model) then
2:     for partition ∈ model.getPartitions() do
3:         if ¬ partition.accept(visitor) then
4:             break
5:         end if
6:     end for
7: end if
8: return visitor.visitOut(model)
```

L'initialisation et la résolution du modèle sont faites au travers des méthodes `initialize` et `solve`. Ces méthodes prennent en paramètre une stratégie de couplage contenant notamment les informations liées au macro pas

de temps ainsi, au couplage des partitions et à la nature du schéma de résolution du couplage. La méthode `solve` est donnée par l'algorithme 8. Ce faisant,

Algorithme 8 `model.solve(strategy)` d'un modèle partitionné.

```
1: hasReachedFinalTime ← strategy.solve(model)
2: model.time += strategy.getEffectiveTimeStep()
3: model.record()
4: return hasReachedFinalTime
```

la résolution du modèle partitionné est entièrement déléguée à la stratégie de couplage. Les stratégies de couplage seront décrites par la suite. Un patron de conception *Stratégie* est utilisé [38]. À la fin de la résolution, le temps du modèle est mis à jour avec le macro pas de temps *effectivement* calculé par les partitions du modèle. Enfin, les sorties du modèle sont enregistrées en appelant la méthode `record` héritée de l'interface `Recordable`.

La classe `PartitionedModel` réalise l'interface `Parameter.DependentInterface` décrite à la section 3.2. Les modèles partitionnés peuvent donc utiliser les fonctionnalités liées au mécanisme de gestion des paramètres. Cependant, celui-ci ne permet que l'échange de paramètres associés à des tableaux de types simples. Par conséquent, un ensemble de classes ont été introduites permettant aux modèles d'utiliser des *variables* associées à des objets Java complexes. Ces classes sont décrites ci-après.

La sauvegarde et la restauration sont possibles en appelant les méthodes `save` et `restore`. Une sauvegarde d'un modèle partitionné est représentée par un objet réalisant l'interface vide `PartitionedModelSave` permettant de conserver l'encapsulation de la classe `PartitionedModel`. En particulier, chaque modèle peut créer sa propre sauvegarde lui permettant d'y inclure des objets qui lui sont propres et de les restaurer. Un patron de conception *Memento* est utilisé [38]. Une réalisation *minimale* de l'interface `PartitionedModelSave` est déjà fournie dans la classes `PartitionedModel`. L'objet de sauvegarde associé contient une copie de la valeur des paramètres et des variables des modèles partitionnés. En interne de cette sauvegarde, les paramètres sont copiés et stockés dans un objet `ParameterProvider` (voir la description des classes de paramètre de la figure 3.6). Dans cette sauvegarde, les variables sont enregistrées par un mécanisme décrit ci-après. Un modèle partitionné peut simplement ajouter des éléments à cette sauvegarde en créant une nouvelle classe implémentant l'interface `PartitionedModelSave`.

3.3.1.2 Les variables des modèles partitionnés

Les classes relatives à la déclaration et aux stockages de variables sont décrites dans le diagramme UML de la figure 3.15.

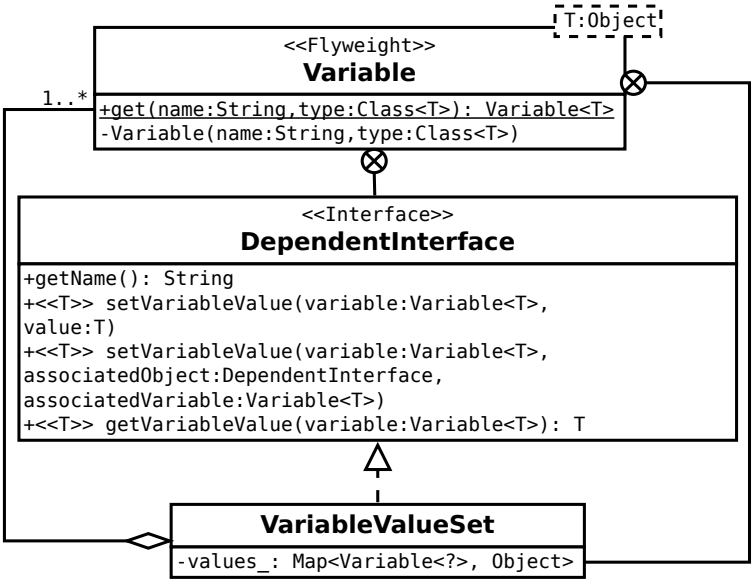


FIGURE 3.15 – Classes relatives aux variables des modèles partitionnés dans l'architecture de couplage.

Représentant une variable, un objet `Variable` a un nom et un type Java générique mais, comme pour les paramètres, ne contient pas l'objet Java associé à la variable et est purement descriptif. Pour optimiser la création des variables, un patron de conception *poids-mouche* [38] est utilisé. Un objet réalisant l'interface `Variable.DependentInterface` déclare un ensemble de variables. Le stockage des valeurs des variables peut être fait dans des objets `VariableValueSet` réalisant l'interface `Variable.DependentInterface`. En interne, les `VariableValueSet` stockent les valeurs des variables dans une `map` Java associant chaque objet `Variable<?>` à l'objet Java représentant la valeur de la variable. Un patron de conception "typesafe heterogeneous container pattern" [10] permet au `VariableValueSet` de stocker des valeurs de types différents, en anglais "heterogeneous", tout en étant à typage sûr, en anglais "typesafe".

3.3.1.3 La sauvegarde des variables

Plusieurs classes ont été intégrées dans l'architecture de couplage pour la sauvegarde automatique des objets Java associés aux variables. Ces classes sont décrites dans le diagramme UML de la figure 3.16.

Pour sauvegarder des variables, un objet `Variable.DependentInterface` peut utiliser un objet `VariableSaver` en appelant sa méthode `save`. Un identifiant unique associé à la sauvegarde dans l'objet `VariableSaver` est donné au modèle partitionné. À partir de cet objet, il peut ensuite restaurer ses variables

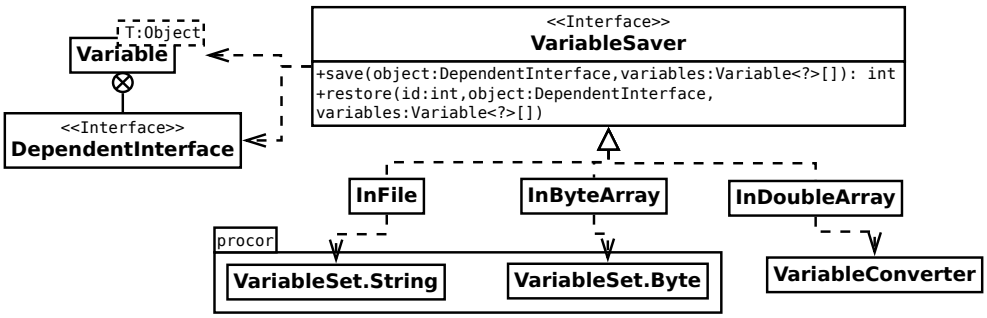


FIGURE 3.16 – Classes relatives à la sauvegarde des variables d'un modèle partitionné.

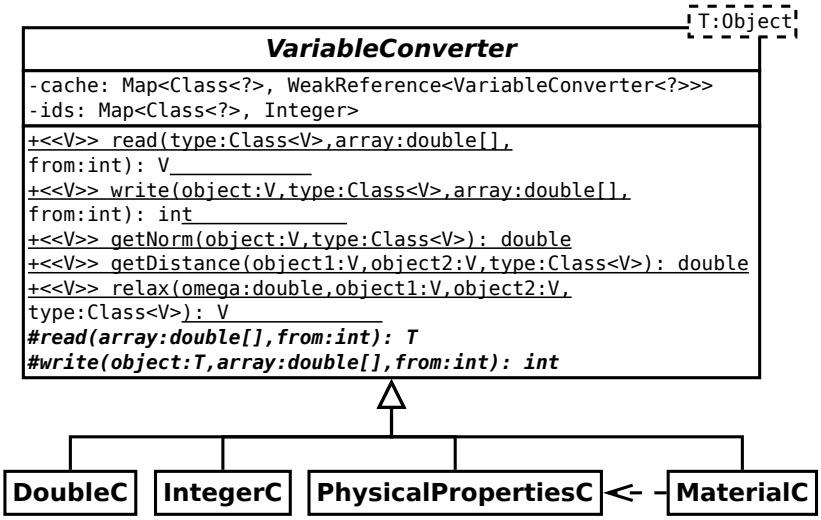


FIGURE 3.17 – Classe permettant la conversion d'objets Java en doubles.

en faisant appel à la méthode `restore` avec cet identifiant.

Différentes réalisations de l'interface `VariableSaver` sont fournies dans l'architecture. Les classes `InByteArray` et `InFile` utilisent toutes les deux le mécanisme de sérialisation de Java transformant les objets en flux d'octets qui est ensuite sauvegarder dans un tableau ou dans des fichiers. Ces deux classes réutilisent les objets `VariableSet` (décrits dans la figure 3.8) de la plate-forme PROCOR pour stocker respectivement le tableau d'octets et le tableau contenant les chemins des fichiers sous forme de chaînes de caractères.

La classe `InDoubleArray` réalise aussi l'interface `VariableSaver`. Elle permet le stockage des objets à sauvegarder dans des tableaux de doubles. Pour la conversion des objets Java en double, elle utilise la classe `VariableConverter` détaillée dans diagramme UML de la figure 3.17. La classe abstraite `VariableConverter<T>` permet la conversion d'objets de type `T`. Elle utilise un patron de conception

Singleton [38]. Une classe héritant de celle-ci peut écrire et lire dans et depuis un tableau des objets Java de type `T`. Quelques implémentations de la classe `VariableConverter<T>` sont données à titre d'exemple. Par exemple, la classe `DoubleC` convertit des objets java de type `Double`, la classe `MaterialC` convertit des objets `PROCOR Material` (représentant, entre autres, un matériau par sa masse, sa température, sa composition et ses propriétés physiques).

L'ensemble de ces implémentations de la classe `VariableConverter<T>` sont privées et contenues dans la classe `VariableConverter`. Seule l'unique instance de la classe `VariableConverter` (car c'est un singleton) peut créée de tels convertisseurs. Les singletons convertisseurs ainsi créés sont stockés dans un cache privé et lié au singleton `VariableConverter` par des références faibles Java. Chaque convertisseur a son unique identifiant, stocké dans la `map ids`. La création des convertisseurs n'est faite que lorsque cela est nécessaire. Par exemple, lors l'appel de la méthode `read` de l'instance convertisseuse retournant un objet de type `V` depuis un tableau de doubles. Naturellement, si l'instance ne trouve pas de convertisseur pour ce type, une exception est levée.

Une liste non exhaustive des fonctionnalités offertes par l'instance convertisseuse est donnée dans le diagramme. Elle peut naturellement écrire un objet dans un tableau par la méthode `write`, opération réciproque de la lecture par la méthode `read`. Elle peut aussi être utilisée pour calculer des distances, des normes ou un barycentre entre deux objets par les méthodes `getDistance`, `getNorm` et `relax`.

3.3.2 Les communications entre modèles

Identifiée précédemment dans l'analyse du besoin de la section 3.2, la stratégie de couplage est l'entité centrale gérant et résolvant le couplage des modèles partitionnés. Elle permet la représentation informatique du problème de couplage décrit par le système d'équations (2.5) et par le graphe de la figure 2.3. Dans les équations $\mathcal{F}_i$ données par

$$\mathcal{F}_i(t^n,\ \mathbf{u_i},\ \{\mathbf{b}_{ji}(\mathbf{u}_j,\ \mathbf{b}_{\star j})\}_{j\in N_{\star i}},\ \mathbf{b}_i) = 0$$

d'un modèle i d'un problème de couplage apparaissent les variables d'interfaces $\mathbf{b}_{ji}(\mathbf{u}_j,\ \mathbf{b}_{\star j})$. Elles correspondent aux variables d'interfaces du modèle i données par les modèle $j \in N_{\star i}$ couplés avec i. Pour les cas les plus simples, lesdites fermetures sont directement utilisées par le solveur.

Cependant, en pratique pour les solveurs existant de la plate-forme, il est fréquent qu'un post-traitement *numérique* soit nécessaire sur une ou plusieurs variables d'interfaces en amont de la résolution *physique*. Dans ces cas, le solveur $\mathcal{M}_i$ peut être vu comme la combinaison d'un solveur physique et de plusieurs traitements numériques, comme décrit dans la figure 3.18). Ce post-traitement des données d'entrées peut par exemple correspondre à des opérations de *prédiction* ou de *correction*, décrites dans la figure 3.13 et expliquée

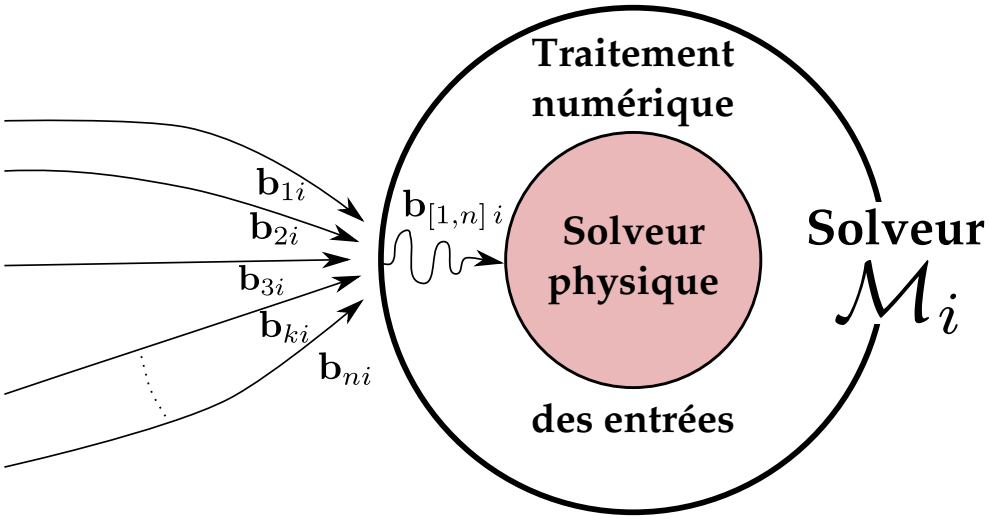


FIGURE 3.18 – Traitement numérique, en amont de la résolution physique, des fermetures données par les modèles couplés avec le modèle i.

dans [35], faites lors de la résolution du couplage par le schéma. Le post-traitement peut aussi correspondre à des opérations de projection liées à la modélisation et/ou la discrétisation numérique du problème physique sous-jacent. Par exemple dans la plate-forme PROCOR,

- c'est le cas lors du calcul d'une température d'interface entre deux domaines différemment maillés nécessitant une *projection* de la température du maillage source vers le maillage destinataire.
- c'est le cas lors du calcul d'un débit de masse en entrée d'un modèle provenant de différents matériaux en fusion d'autres modèles nécessitant une *opération* globalisant les différents matériaux sources.
- c'est le cas lors du calcul d'un flux de chaleur *reçu* de différentes sources qui est ensuite *projeté* et *envoyé* sur différents domaines, permettant ainsi une conservation de l'énergie associée à ce flux.

Généralement, ces traitements se divise en trois étapes successives : une *communication* de réception des données d'entrées, puis un ensemble d'*opérations* sur ces données, et enfin une *communication* d'envoi des données post-traitées. De ce fait, un *communicateur* entre modèles sera composé de deux communications, permettant la réception et l'envoi de données, et d'un ensemble d'opérations. Un communicateur peut être réutilisé plusieurs fois par un ensemble

$$I \overset{\text{def.}}{=} [1, m]$$

de solveurs physiques, par exemple les flux de chaleur envoyés vers plusieurs modèles, et a donc m sorties. Il calcul ses sorties à partir de variables d'inter-

faces données par un ensemble

$$J \overset{\text{def.}}{=} [1, n]$$

de modèles couplés avec les m modèles de sortie. Mathématiquement, les communications et opérations d'un communicateur peuvent être décrites par des fonctions mathématiques :

1. une communication de réception depuis les m modèles d'entrées des variables d'interfaces est notée
$$\mathcal{R}_J(\{\mathcal{M}_j\}_{j\in J}) \overset{\text{def.}}{=} \{\mathbf{b}_{ji}(\mathbf{u}_j,\ \mathbf{b}_{\star j})\}_{j\in J, i\in I}. \tag{3.2}$$
2. une ou des opérations post-traitant ces variables d'interfaces notées
$$\mathcal{P}_{JI}(\{\mathbf{b}_{ji}(\mathbf{u}_j,\ \mathbf{b}_{\star j})\}_{j\in J, i\in I}) \overset{\text{def.}}{=} \mathbf{b}_{JI}. \tag{3.3}$$
3. une opération d'envoi des données post-traitées vers les modèles de sorties est notée
$$\mathcal{E}_I(\mathbf{b}_{JI}) \overset{\text{def.}}{=} \{\mathbf{b}_{Ji}\}_{i\in I}. \tag{3.4}$$

Le composition des équations (3.2), (3.3) et (3.4) permet la réception, le post-traitement et l'envoi des données depuis l'ensemble J de modèles d'entrées et vers l'ensemble I de modèles de sortie. Le *communicateur* C_{JI} entre les n modèles d'entrées et les m modèles de sorties est défini par

$$C_{JI} \overset{\text{def.}}{=} \mathcal{E}_I \circ \mathcal{P}^l_{JI} \circ \ldots \circ \mathcal{P}^1_{JI} \circ \mathcal{R}_J(\{\mathcal{M}_j\}_{j\in J}) = \{\mathbf{b}_{Ji}\}_{i\in I}. \tag{3.5}$$

3.3.2.1 La classe des communicateurs

Dans l'architecture, un communicateur est représenté par la classe `Communicator` décrite dans le diagramme UML de la figure 3.19. Il est complètement défini par la donnée des variables d'interfaces reçues de l'équation (3.2), la donnée des opérations définies par l'équation (3.3) et la donnée des variables de sorties dans l'équation (3.4).

Les différentes méthodes d'un communicateur de l'architecture permettent d'une part aux modèles d'échanger des données correctement projetées et d'autre part aux schémas de couplage de manipuler les données comme ils le souhaitent (voir figure 3.13) :

les méthodes `receive` et `send` permettent de recevoir et d'envoyer des données depuis et vers un modèle.

la méthode `predict` permet la prédiction des variables d'entrées. Différentes stratégies de prédiction permettent, à partir de valeurs des variables aux instants précédents, de prédire une valeur plus ou moins exacte de la variable au macro pas de temps suivant, voir [83] pour des exemples de prédicteurs. Cette méthode utilise l'objet `predictor` donné à la construction du communicateur.

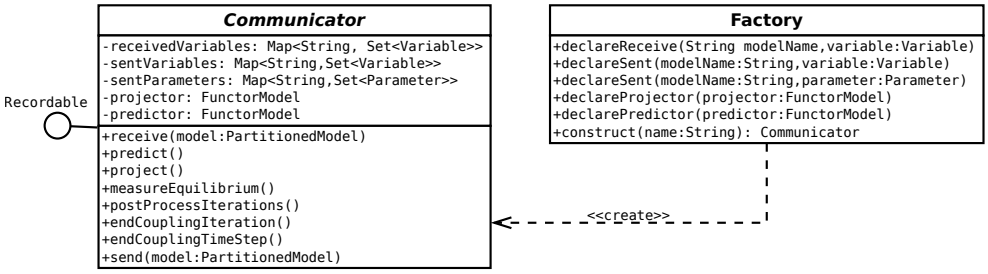


FIGURE 3.19 – Classe représentant les communicateurs permettant aux modèles partitionnés l'échange de données.

la méthode `project` permet la projection des variables des n modèles d'entrées vers les m modèles de sorties. Cette méthode utilise l'objet `projector` donné à la construction du communicateur.

la méthode `measureEquilibrium` permet la mesure du déséquilibre à l'interface entre les n modèles d'entrées et les m modèles de sortie. Elle calcule le résidu $\mathcal{R}$ donné par l'équation (2.11).

la méthode `postProcessIteration` permet le post-traitement des valeurs obtenues à la fin d'un macro pas de temps. Par exemple la correction ou la moyenne lors d'un SCE, ou la relaxation ou un post-traitement plus complexe lors d'un SCI.

les méthodes `endCouplingIteration` et `endCouplingTimeStep` permettent d'informer le communicateur de la fin d'un calcul sur une itération ou un macro pas de temps et sont généralement utilisées pour réinitialiser des valeurs ou les rendre nulles pour débuter une nouvelle itération ou un nouveau macro pas de temps.

Les informations concernant les variables reçues des n modèles d'entrées et les variables et/ou paramètres envoyés aux m modèles de sorties sont stockées en interne du communicateur dans les objets `receivedVariables`, `sentVariables` et `sentParameters`. De même pour les différentes opérations de post-traitement telles que la prédiction des variables d'entrées ou la projection des variables de sortie. L'ensemble de ces informations sont données à la construction du communicateur, rendant ladite construction complexe. Pour remédier à ce problème, un objet `Factory` permet la construction simple de communicateurs. La construction du communicateur se fait suite à la donnée de :

- L'ensemble des variables d'entrées $\{\mathbf{b}_{ji}(\mathbf{u}_j, \mathbf{b}_{\star j})\}_{j\in J, i\in I}$ données par les solveurs $\{\mathcal{M}_j\}_{j\in J}$.
- L'ensemble des variables de sorties $\{\mathbf{b}_{Ji}\}_{i\in I}$ envoyées vers les solveurs $\{\mathcal{M}_i\}_{i\in I}$.
- Un ensemble d'opérateurs, représentés par la classe `ParameterizedFunctor`, décrivant les différentes opérations $\mathcal{P}_{JI}$ de projection, prédiction, etc.

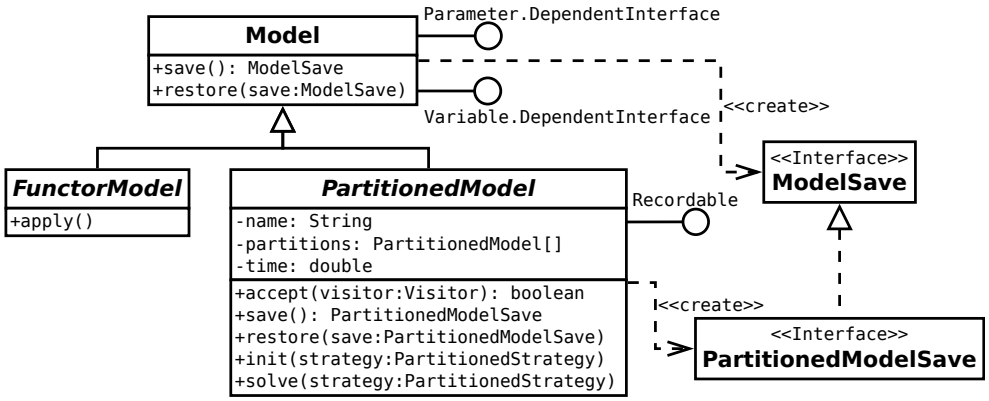


FIGURE 3.20 – Classe `FunctorModel` représentant les opérateurs de projection et prédiction utilisés dans les communicateurs.

Différentes implémentations de cette factory permettent de construire différents types de communicateurs. Un patron de conception *Factory* est utilisé [38]. Il est par exemple possible de créer des communicateurs à partir d'un projecteur, avec ou sans prédicteur, ou de construire des communicateurs sans projection, appelés `IdentityCommunicator`, liant directement les entrées aux sorties.

Les objets de projection `projector` et de prédiction `predictor` sont représentées dans l'architecture par des objets de type `FunctorModel`, décrit dans le diagramme UML de la figure 3.20. Pour introduire ces objets, la classe `Model` a été ajoutée dans l'architecture. Elle permet la déclaration de variables et de paramètres en implémentant les interfaces `Variable.DependentInterface` et `Parameter.DependentInterface`. Elle permet aussi la sauvegarde et la restauration de ses variables et paramètres par les méthodes `save` et `restore`. C'est de cette classe qu'héritent les modèles partitionnés `PartitionedModel`. Les opérations de projection et prédiction sont représentées par des objets `FunctorModel` qui héritent de la classe `Model` et ayant en plus une méthode abstraite `apply` permettant un ensemble d'opérations. Héritant de la classe `Model`, ces opérateurs peuvent déclarés des variables et paramètres. Ainsi un communicateur peut donner à son objet `projector` ou `predictor` les variables qu'il a reçu et récupérer les variables projetées ou prédites pour les envoyer à ses modèles de sortie.

Enfin, la classe `Communicator` réalisent l'interface `Recordable`. Il est ainsi possible d'enregistrer les données transitant par un communicateur, par exemple les résidus aux interfaces entre les modèles d'entrées et de sorties, ou les différentes itérations des variables d'interfaces communiquées par le communicateur.

Par exemple, le communicateur C_{JI} donné par précédemment dans l'équa-

tion (3.5) peut être défini dans l'architecture par un objet `Communicator`. La méthode `receive` représente l'opérateur mathématique de réception $\mathcal{R}_J$, la méthode `send` représente l'opérateur d'envoi $\mathcal{E}_I$, et la méthode `project` ainsi que l'objet `projector` représentent un opérateur de post-traitement de projection $\mathcal{P}_{JI}$. Avec ce communicateur, la réception, une opération de projection et l'envoi de données est donnée par l'algorithme 9.

Algorithme 9 Réception, projection et envoi de données avec un objet `communicator` $\equiv C_{JI}$.

1: **for** $j \in J$ **do**
2: `communicator.receive(`$\mathcal{M}_j$`) /* Réception depuis` $\mathcal{M}_j$ `*/`
3: **end for**
4: `communicator.project() /* Projection des données */`
5: **for** $i \in I$ **do**
6: `communicator.send(`$\mathcal{M}_i$`) /* Envoi vers` $\mathcal{M}_i$ `*/`
7: **end for**

3.3.2.2 Les effets de bord entre modèles et communicateurs

Initialement, les variables envoyées par les modèles d'entrées étaient directement données au communicateur. Celui-ci manipulait ensuite ces variables et envoyait ses données à ses modèles de sorties. Des objets Java envoyés par les modèles d'entrées pouvaient alors être modifiés lors des communications provoquant des *effets de bord* entre modèles et communicateurs. Pour remédier à ce problème, un mécanisme de sauvegarde et de copie des variables d'entrées à été introduit dans les communicateurs. Ceux-ci peuvent alors être vus comme des *tampons* entre les modèles et plus aucun effet de bord n'est possible entre les modèles et les communicateurs.

Le schéma de la figure 3.21 décrit de tels communicateurs *tampons* et leur représentation informatique dans l'architecture. À chaque modèle partitionné d'entrées $\mathcal{M}_i$ est associé un objet `Model` $\tilde{\mathcal{M}}_i$ (objets décrits dans le diagramme de la figure 3.20). Cet objet est construit à partir de l'ensemble des variables reçues $\mathbf{b}_{ji}$ depuis le modèle d'entrées $\mathcal{M}_j$ pour $j \in J$.

À la réception des variables du modèle partitionné $\mathcal{M}_i$ d'entrées, les valeurs des variables, qui peuvent être des objets Java complexes, sont données au modèle $\tilde{\mathcal{M}}_i$ associé. Le modèle $\tilde{\mathcal{M}}_i$ est ensuite sauvegardé en faisant appelle à sa méthode `save`. En interne de cette méthode, les valeurs de variables sont converties (en doubles ou bits avec le mécanisme Java de sérialization) et copiées à l'identique à l'aide d'objets `VariableSaver` (un mécanisme de sauvegarde créé pour la thèse décrit précédemment dans le diagramme de la figure 3.17).

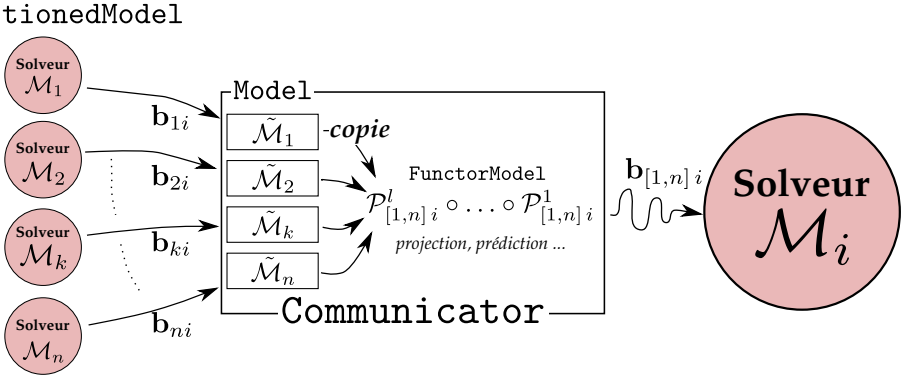


FIGURE 3.21 – Représentation informatique des communicateurs permettant d'éviter les effets de bord sur les variables d'entrées qui sont converties et copiées dans des *tampons* d'entrées.

Ensuite, lorsque le communicateur veut manipuler les valeurs reçues par le modèle partitionné pour des opérations $\mathcal{P}_{[1,n]\,i}$ décrites par des `FunctorModel`, le modèle associé est restauré par sa méthode `restore` et des objets Java identiques à ceux reçus sont recréés puis données aux opérateurs. Ceux-ci ne manipulent donc jamais les objets initialement reçus des modèles mais toujours des copies.

3.3.3 Les stratégies de couplage

Une stratégie de couplage d'un modèle partitionné permet :
— d'une part la représentation des couplages entre partitions du modèles.
— d'autre part la résolution de ce couplage.

Elle a une structure d'arbre *équivalente* au modèle partitionné qu'elle résout. En particulier, chaque niveau de l'arbre contient un ensemble de stratégies de couplage permettant la résolution des partitions du modèle partitionné père commun à toutes les partitions. Ces stratégies sont connectées par un ensemble de communicateurs permettant aux partitions associées aux stratégies d'échanger des données lors de la résolution. Par conséquent, chaque niveau de l'arbre est un graphe *orienté* composé de stratégies et de communicateurs.

Un exemple d'arbre est donné à la figure 3.22. Dans cet arbre, une stratégie de couplage $\mathcal{S}$ est définie. Elle est associée à un modèle partitionné $\mathcal{M}$ et permet sa résolution. La stratégie $\mathcal{S}$ est composée de k sous stratégies de couplage $\mathcal{S}_1, \ldots, \mathcal{S}_k$ associées aux partitions $\mathcal{M}_1, \ldots, \mathcal{M}_k$ du modèle $\mathcal{M}$. Les stratégies sont connectées par les communicateurs $C_{[2,3,4]\,1}$, $C_{4,3}$ et $C_{[1,4]\,k}$. Pour résoudre le couplage des partitions, les sous stratégies sont elles-mêmes com-

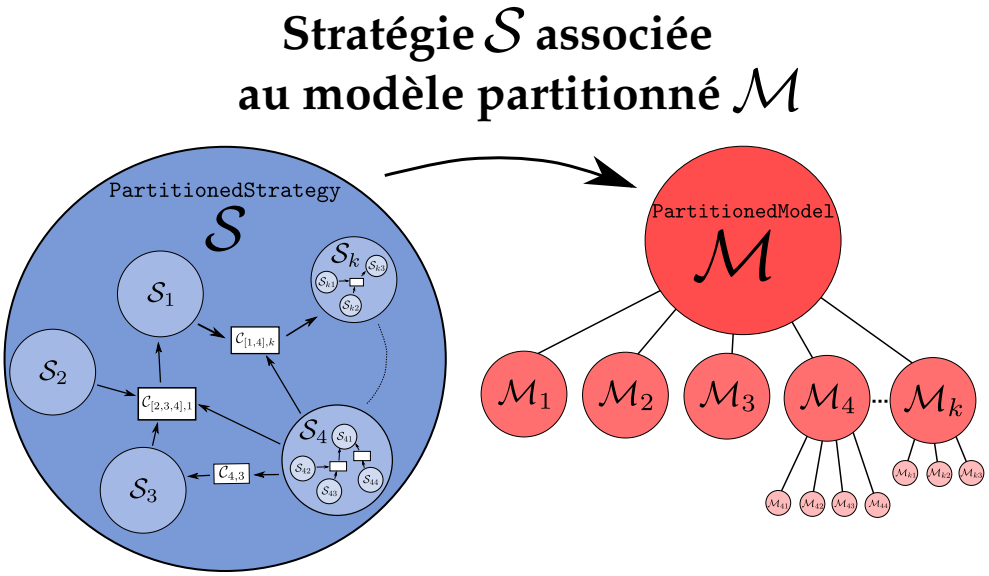


FIGURE 3.22 – Stratégie de couplage $\mathcal{S}$ d'un modèle $\mathcal{M}$ formant un arbre de stratégies. Chaque niveau de l'arbre correspond à un graphe composé de stratégies et de communicateurs.

posées de sous-stratégies et de communicateurs, par exemple la stratégie $\mathcal{S}_k$ composée de $\mathcal{S}_{k1}$, $\mathcal{S}_{k2}$, $\mathcal{S}_{k3}$ et d'un communicateur.

Les stratégies sont connectées à un ensemble de communicateurs. Ces communicateurs peuvent être des communicateurs d'entrées ou de sorties de la partition associée à la stratégie. De ce fait, une partition est couplée à une autre s'il existe un chemin dans le graphe partant de la partition et arrivant à l'autre partition.

La résolution du couplage donnée par la stratégie correspond à un parcours du graphe des stratégies et communicateurs du graphe. Il peut passer plusieurs fois par les mêmes stratégies et les mêmes communicateurs. La stratégie de résolution dicte la manière dont les données envoyées par les partitions transitent dans les communicateurs. En particulier, c'est la stratégie qui va résoudre chaque partition, en appelant leur méthode `solve(strategy)`, et donner l'instant durant la résolution auquel les données des partitions doivent être envoyées aux communicateurs et auquel les données des communicateurs doivent être envoyées aux partitions, en appelant respectivement les méthodes `receive(partition)` et `send(partition)`.

Pour représenter de telles stratégies dans l'architecture, des classes spécifiques ont été créées dans l'architecture. Elles sont présentées ci-après.

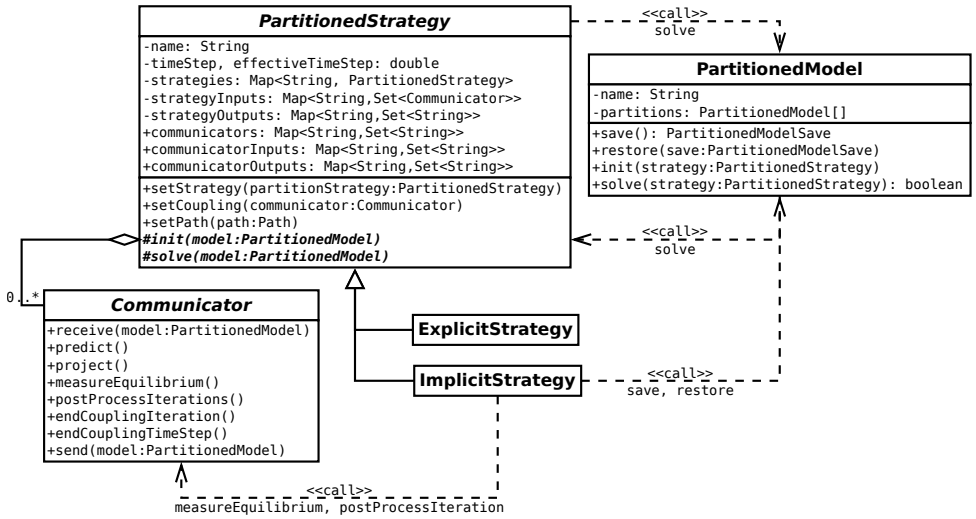


FIGURE 3.23 – Classe représentant les stratégies de couplage des modèles partitionnés.

3.3.3.1 La classe des stratégies

Une stratégie de couplage est représentée dans l'architecture par un objet de la classe abstraite `PartitionedStrategy`, décrite dans le diagramme UML de la figure 3.23. Celle-ci est construite à partir d'un ensemble d'objets `PartitionedStrategy` correspondant à ses sous stratégies (par la méthode `setPartitionedStrategy`) et d'un ensemble d'objets `Communicator` correspondant aux communicateurs entre partitions (par la méthode `setCommunicator`). En interne de la classe, le graphe orienté est représenté par la liste de ses nœuds, les stratégies et communicateurs, et la liste d'adjacence de ceux-ci. Dans cette liste est stockée l'ensemble des connexions entre stratégies et communicateurs. Une stratégie, respectivement un communicateur, sera connectée en entrées et en sorties à un ensemble de communicateurs, respectivement un ensemble de stratégies. Les nœuds du graphe (les stratégies et communicateurs) sont stockés respectivement dans les attributs `strategies` et `communicators`. Ils sont entièrement identifiés par leur nom. Les deux listes d'adjacences des nœuds (les stratégies et communicateurs) en entrées et sorties sont stockées dans les attributs `inputs` et `outputs`.

Viennent ensuite un ensemble de fonctionnalités liées à la résolution du couplage. La méthode `setTimeStep` permet de définir le macro pas de temps de résolution du couplage. La méthode `getEffectiveTimeStep` retourne le macro pas de temps effectivement calculer après la résolution (qui peut être plus faible si un événement stoppant la résolution a été détecté). La méthode `setPath` permet de définir le chemin dans les stratégies du graphe (identifiées par leur nom) parcouru par la stratégie lors de la résolution. Enfin, d'autres

fonctionnalités relatives à la résolution numérique non présentées dans la diagramme sont offertes par la classe `PartitionedStrategy`, par exemple le *sous-cyclage* d'une ou d'un ensemble de partitions.

Étant donné un modèle partitionné, une stratégie de couplage résout les partitions et les fait communiquer suivant l'ordre défini par le chemin dans le graphe. Une classe héritant de la classe abstraite `PartitionedStrategy` doit implémenter les méthodes abstraites `init` et `solve`, initialisant et résolvant le couplage des partitions (décrites par la suite). Différentes stratégies existent déjà dans l'architecture et correspondent à différents schémas de résolution du couplage. Naturellement, d'autres schémas sont *implémentables* dans l'architecture en créant de nouvelles stratégies héritant de la classe `PartitionedStrategy`. Les deux classes de stratégies présentes actuellement dans l'architecture sont :

— la classe `ExplicitStrategy` permet une résolution semi explicite du couplage par un SCE de type CSS (voir la section 2.1.6).
— la classe `ImplicitStrategy` permet une résolution implicite et synchronisée du couplage par un SCI (voir la section 2.1.7 et la section 2.2). Avec cette stratégie, il est possible d'utiliser tous les schémas implicites présentés précédemment, par exemple Gauss-Seidel par blocs, Gauss-Seidel par blocs avec relaxation ou des méthodes de Quasi-Newton.

Ci-après sont présentées ces deux classes et en particulier l'implémentation logicielle des algorithmes de résolution associés aux schémas de couplage. La méthode `init`, pour l'instant commune aux deux classes, est présentée en première. On présente ensuite la méthode `solve`, naturellement avec une implémentation différente pour les deux classes.

3.3.3.2 L'initialisation des stratégies de couplage

On décrit ci-après l'implémentation de la méthode `init` dans les classes `ImplicitStrategy` et `ExplicitStrategy`. La méthode `init` des stratégies de couplage permet l'initialisation d'une part des partitions du modèle partitionné et d'autre part du couplage entre les partitions. La stratégie d'initialisation doit être cohérente avec ce que les partitions attendent comme données d'entrées pour pouvoir être correctement initialisé. Si cela n'est pas le cas, certaines partitions ne pourront être initialisées et l'initialisation du modèle partitionné retournera une erreur. Les données nécessaires à l'initialisation des partitions peuvent être différentes des données nécessaires lors des cycles du calcul.

Lorsque la stratégie de couplage a terminé son initialisation, les partitions et le couplage entre ces partitions sont initialisés à l'instant t^0.

Le pseudo code de la méthode telle qu'elle est implémentée dans les classes `ImplicitStrategy` et `ExplicitStrategy` est donné par l'algorithme 10 et est

détaillé ci-après. Ligne 1 de l'algorithme 10, l'algorithme parcourt les stra-

Algorithme 10 Pseudo code de `init(PartitionedModel)`

```
1:  for strategy ∈ path do
2:      partition ← model.getPartition(strategy.name))
3:      for communicator ∈ inputs.get(strategy.name) do
4:          for coupledPartition ∈ inputs.get(strategy.name) do
5:              if coupledPartition est initialisée then
6:                  communicator.receive(coupledPartition)
7:              end if
8:          end for
9:          communicator.send(partition)
10:     end for
11:     partition.init(strategy)
12:     for communicator ∈ outputs.get(strategy.name) do
13:         communicator.receive(partition)
14:     end for
15: end for
16: for communicator ∈ strategy.communicators do
17:     communicator.endCouplingTimeStep()
18: end for
```

tégies du graphe suivant le chemin `path` défini dans la stratégie de couplage mère. À la ligne 2, la partition du modèle associée à la sous stratégie `strategy` de la stratégie mère est stockée dans la variable `partition`. Ensuite, l'algorithme se divise en trois étapes :

1. De la ligne 3 à la ligne 11, les communicateurs `communicator` liés aux entrées de la stratégie envoient leurs données. Si les partitions en entrées de ces communicateurs n'ont pas encore été initialisées, aucune donnée n'a été envoyée. Dans ce cas, de la ligne 4 à 9, des données évaluées à l'instant t^0 sont reçues de ces partitions.
2. À la ligne 12, la partition est initialisée à l'instant t^0.
3. De la ligne 13 à 15, les communicateurs en sorties de la partition reçoivent de la partition les données initialisées à l'instant t^0.

Enfin, de la ligne 17 à 19, l'algorithme informe les communicateurs de la stratégie de la fin d'un macro pas de temps.

3.3.3.3 La résolution du couplage dans la classe `ExplicitStrategy`

Le schéma de couplage semi-explicite CSS est implémenté dans la méthode `solve` de la classe `ExplicitStrategy`. Cette méthode permet la résolution du couplage des partitions d'un modèle `model` sur un macro pas de temps entre les instants t^n et $t^{n+1} = t^n + \Delta t$. Au début de la méthode, il est supposé que les partitions et leurs valeurs sont toutes évaluées à l'instant t^n.

Le pseudo code de la méthode `solve` est donné par l'algorithme 11 et est détaillé ci-après. Ligne 1 de l'algorithme 11, l'algorithme parcourt les stra-

Algorithme 11 Pseudo code de `ExplicitStrategy.solve`

```
1:  for strategy ∈ path do
2:      partition ← model.getPartition(strategy.name))
3:      for communicator ∈ inputs.get(strategy.name) do
4:          for coupledPartition ∈ inputs.get(strategy.name) do
5:              if coupledPartition.getTime() == t^n then
6:                  communicator.receive(coupledPartition)
7:                  communicator.predict(coupledPartition)
8:              end if
9:          end for
10:         communicator.send(partition)
11:     end for
12:     partition.solve(strategy)
13:     for communicator ∈ outputs.get(strategy.name) do
14:         communicator.receive(partition)
15:     end for
16: end for
17: for communicator ∈ strategy.communicators do
18:     communicator.endCouplingTimeStep()
19: end for
```

tégies du graphe suivant le chemin `path` défini dans la stratégie de couplage mère. À la ligne 2, la partition du modèle associée à la sous stratégie `strategy` de la stratégie mère est stockée dans la variable `partition`. Ensuite, l'algorithme se divise en trois étapes :

1. De la ligne 3 à la ligne 11, les communicateurs `communicator` liés aux entrées de la stratégie envoient leurs données. Si les partitions en entrées de ces communicateurs n'ont pas encore été résolues, aucune donnée n'a été envoyée. Dans ce cas, de la ligne 4 à 9, des données sont reçues de ces partitions et prédites à l'instant t^{n+1}.
2. À la ligne 12, la partition est résolue sur un macro pas de temps, avec ou sans sous-cyclage.
3. De la ligne 13 à 15, les communicateurs `communicator` en sorties de la partition reçoivent de la partition les données nouvellement calculées et évaluées à t^{n+1}.

Enfin, de la ligne 17 à 19, l'algorithme informe les communicateurs de la stratégie de la fin d'un macro pas de temps.

La stratégie `ExplicitStrategy` correspond à la résolution du couplage des partitions par chaînage des partitions. En la construisant avec les bons

communicateurs, elle doit permettre de retrouver les résultats donnés par le chaînage des modèles dans la plate-forme d'avant thèse.

Dans la résolution de la stratégie `ExplicitStrategy` (dans sa méthode `solve`), aucun mécanisme de synchronisation n'est intégré car il serait trop limité. Au cours d'un macro pas de temps, il serait par exemple possible de s'arrêter dés qu'une partition détecte un événement, de restaurer toutes les partitions résolues avant celle-ci, puis de relancer la résolution. Néanmoins, il n'est pas sur que l'événement ré-apparaisse lors de la seconde résolution, ou qu'il apparaisse au même instant. Très rapidement, la stratégie de synchronisation en gardant un schéma de résolution CSS devient compliquée et itérative (et commence à ressembler à une stratégie de résolution implicite, présentée ci-après).

3.3.3.4 La résolution du couplage dans la classe `ImplicitStrategy`

Le schéma de couplage implicite (donné par l'algorithme 4) et synchronisé (donné par l'algorithme 5) est implémenté dans la méthode `solve` de la classe `ImplicitStrategy`. Cette méthode résout le couplage des partitions d'un modèle `model` sur un macro pas de temps entre les instants t^n et $t^{n+1} = t^n + \Delta t$. Au début de la méthode, il est supposé que les partitions et leurs valeurs sont toutes évaluées à l'instant t^n.

Le pseudo code de la méthode `solve` est donné par l'algorithme 12 et est détaillé ci-après. L'algorithme de résolution implicite étant plus élaboré que

Algorithme 12 Pseudo code de `ImplicitStrategy.solve`

```
1: Sauvegarder les partitions à t^n.
2: k ← 1
3: Δt^k ← Δt^objectif
4: while non convergence do
5:     Calculer une itération de couplage des partitions entre
       t^n et t^n + Δt^k (alg.13).
6:     Mesurer la convergence aux interfaces entre les
       partitions et appliquer le post-traitement sur les
       itérations (alg.14).
7:     Mesurer la convergence des macro pas de temps
       effectivement calculés par les partitions (alg.15).
8:     if non convergence then
9:         Restaurer les partitions.
10:        Δt^{k+1} ← Δt calculé par alg.15.
11:        k ← k + 1
12:    end if
13: end while
14: Informer les partitions de la fin du pas de temps.
```

l'algorithme de résolution explicite 11, il est divisé en sous algorithmes détaillés ci-après si nécessaire. Pour pouvoir faire des retours au début du macro pas de temps de couplage, les modèles sont sauvegardés à l'instant t^n à la ligne 1. Les variables k et Δt^k d'itérations sont initialisées ligne 2 et 3. Ensuite, l'algorithme itère tant qu'une certaine convergence n'a pas été atteinte :

1. Ligne 5, une itération de couplage est calculée par l'algorithme 13.
2. À la fin de l'itération, ligne 6, les déséquilibres aux interfaces entre partitions est mesuré par l'algorithme 14. Si ces déséquilibres sont trop importants par rapport à la tolérance du schéma, le schéma de couplage n'a pas convergé. Dans ce cas et si c'est nécessaire, le post-traitement (par exemple une relaxation) sur les itérations est appliqué.
3. Ligne 7, la convergence sur la détection de l'instant d'un éventuel événement déclenché par l'une des partitions est mesuré par l'algorithme 15.
4. Ligne 8 à 11, si le critère de convergence n'est pas satisfait, les partitions sont restaurées à l'instant t^n, et les itérés sont mis à jour.

Pour chaque partition parcourue dans l'ordre du chemin `path` de la stratégie, l'algorithme 13 calculant une itération de couplage entre les partitions peut se diviser en trois étapes. De la ligne 3 à 13, les communicateurs associés

Algorithme 13 Calcul d'une itération de couplage des partitions dans la méthode `ImplicitStrategy.solve`.

```
1:  for strategy ∈ path do
2:      partition ← model.getPartition(strategy))
3:      for communicator ∈ inputs.get(strategy) do
4:          if k < 2 then
5:              for coupledPartition ∈ inputs.get(communicator) do
6:                  if coupledPartition.getTime() == t^n then
7:                      communicator.receive(coupledPartition)
8:                      communicator.predict(coupledPartition)
9:                  end if
10:             end for
11:         end if
12:         communicator.send(partition)
13:     end for
14:     partition.solve(strategy)
15:     for communicator ∈ outputs.get(strategy) do
16:         communicator.receive(partition)
17:     end for
18: end for
```

aux entrées de la partition courante envoient les données évaluées à l'instant

t^{n+1}. Si c'est la première itération, ligne 4, il est possible que les données soient évaluées à t^n auquel cas les données sont prédites par le communicateur. Ensuite, ligne 14, la partition est résolue entre l'instant t^n et $t^n + \Delta t^k$. Si un événement apparaît, elle ne calcule pas le macro pas de temps complet et informe la stratégie du macro pas de temps qu'elle a *effectivement* calculé. Enfin, de la ligne 15 à 17, la partition envoie ses données à ses communicateurs de sorties.

Pour chaque partition parcourue dans l'ordre du chemin `path` de la stratégie, l'algorithme 14 mesure les déséquilibres aux interfaces entre partitions et post-traite les itérations. Pour ce faire, il parcourt l'ensemble des commu-

Algorithme 14 Test de convergence et post-traitement des itérations de couplage dans la méthode `ImplicitStrategy.solve`

```
1:  for strategy ∈ path do
2:      partition ← model.getPartition(strategy)
3:      for communicator ∈ outputs.get(strategy) do
4:          for coupledPartition ∈ outputs.get(communicator) do
5:              if coupledPartition résolue avant partition then
6:                  communicator est implicite
7:              end if
8:          end for
9:          if communicator est implicite then
10:             ε_rel.^{Γ,k} ← communicator.measureEquilibrium()
11:             if ε_rel.^{Γ,k} > ε_rel. then
12:                 communicator.postProcessIteration()
13:             end if
14:         end if
15:     end for
16: end for
```

nicateurs de sorties de la partition. Il faut ensuite savoir si ce communicateur correspond à une communication implicite. Une communication entre n partitions d'entrées et m partitions de sorties est dite implicite si au moins l'une des partitions de sorties a été résolue avant l'une des partitions d'entrées. C'est ce qui est testé de la ligne 4 à 8. Si le communicateur est implicite, ligne 9, le déséquilibre relatif à l'interface $\epsilon_{\text{rel.}}^{\Gamma,k}$ est calculé et est comparé à la tolérance sur le couplage $\epsilon_{\text{rel.}}$ de la stratégie. Si le déséquilibre est trop important par rapport à cette tolérance, le post-traitement *spécifié* et *encapsulé* dans la communicateur est effectué ligne 12.

Enfin, l'algorithme 15 interroge les partitions pour récupérer les macro pas de temps qu'elles ont effectivement calculés, ligne 1. Ensuite, l'algorithme correspond exactement à celui décrit dans la section 2.2 sur la détection des événements. À la fin de son exécution, un nouveau macro pas de temps de

Algorithme 15 Test de convergence de Δt^k et calcul du nouveau macro pas de temps Δt^{k+1} dans la méthode `ImplicitStrategy.solve`.

$\tilde{\Delta t}^k \leftarrow$ `le plus petit macro pas de temps effectivement calculé par les partitions.`
if $\tilde{\Delta t}^k < \Delta t^k$ **then**
    $\epsilon_{\text{rel.}}^{\Delta t,k} \leftarrow |\tilde{\Delta t}^k - \Delta t^k| / \tilde{\Delta t}^k$
    if $\epsilon_{\text{rel.}}^{\Delta t,k} < \epsilon_{\text{rel}}^{\Delta t}$ **then**
        $\Delta t^{k+1} \leftarrow \tilde{\Delta t}^k$
        `la stratégie a convergé sur la détection de l'événement.`
    else
        $\Delta t^{k+1} \leftarrow \alpha \tilde{\Delta t}^k + (1-\alpha)\Delta t^k$
    end if
else if $\tilde{\Delta t}^k < \Delta t$ **then**
    $\Delta t^{k+1} \leftarrow \beta \tilde{\Delta t}^k + (1-\beta)\Delta t$
end if

couplage Δt^{k+1} est calculé.

Remarque sur l'implémentation actuelle de la méthode `solve` dans la classe `ImplicitStrategy`. L'implémentation actuelle est très générique et permet à priori d'utiliser tous les schémas de couplage implicites. Comme expliqué dans la section 2.1.7 sur les SCI, la principale différence entre les différents schémas réside dans la nature du post-traitement fait à la fin des itérations sur les variables de couplage. À l'interface entre plusieurs partitions, le post-traitement de la ou des variables d'interface est encapsulé dans le communicateur associé à cette interface. Il suffit donc de changer le communicateur ou simplement l'implémentation de la méthode `postProcessIteration` pour changer la nature du schéma implicite. Il suffit ensuite de créer une stratégie `ImplicitStrategy` avec le ou les nouveaux communicateurs pour avoir une nouvelle stratégie implicite.

Chapitre 4

Problèmes couplés de la plateforme PROCOR

L'objectif de ce chapitre est la résolution numérique de problèmes couplés à l'aide des schémas de couplage et de synchronisation présentés chapitre 2 et implémentés dans la nouvelle architecture de couplage présentée chapitre 3. Chaque section de ce chapitre concerne un problème de couplage issu de la plate-forme PROCOR : **la conduction de la chaleur dans plusieurs domaines couplés** (section 4.1), **la vidange d'une couche par plusieurs débits de masse couplés** (section 4.2) et **le couplage des modèles d'une application industrielle PROCOR** (section 4.3).

Problème de conduction de la chaleur dans deux domaines couplés. Le partitionnement correspond à un partitionnement topologique de l'espace en plusieurs domaines couplés dans lesquels on modélise la conduction de la chaleur. Dans cette section, on se concentre sur une modélisation largement utilisée dans la plate-forme PROCOR : la modélisation *point* ou *0D* ou *"lumped parameter"* (LP). On décrit pour ce formalisme les équations différentielles de conservation de la masse et de l'énergie ainsi que leurs équations de fermetures permettant de décrire l'évolution de chaque sous domaine. Entre deux domaines couplés, différentes interfaces seront considérées : une interface fixe avec continuité du flux de chaleur et de la température ou une interface mobile correspondant à une condition de Stefan (voir [70]). Le changement du type d'interface, créant une forte discontinuité dans les équations, correspond à un événement déclenché par un modèle sur lequel le schéma doit se synchroniser. La résolution du problème couplé avec des modèles "lumped parameter" (LP) sera comparée à une résolution avec des modèles maillés en espace. Largement utilisée dans le contexte AG et dans d'autres domaines, on mettra ainsi en valeur l'impact sur le couplage de la modélisation 0D par rapport à une modélisation maillée et les solutions apportées par les schémas de couplage et de synchronisation proposés. Une analyse numérique permettra de

définir des critères de stabilité des SCE et de convergence des SCI qui seront confirmés par les résultats numériques. On verra en particulier qu'il est possible d'*accélérer* les itérations d'un SCI pour le rendre aussi performant qu'un SCE avec un macro pas de temps de couplage petit pour lequel la résolution explicite du couplage est précise. Cette section est liée à l'annexe A dans laquelle est décrite l'intégration des équations sur les domaines permettant de construire les modèles "lumped parameter". Enfin, cette section a fait l'objet d'une publication (en relecture) dans *"International Journal for Multiscale Computational Engineering"* et est retranscrite telle qu'elle.

Problème de la vidange d'une couche par des débits couplés. On s'intéresse aux couplages entre différents modèles calculant des débits de masse entrants ou sortants d'une couche d'un matériau. Ces débits gouvernent l'évolution du matériau au cours du temps. Un événement important sur lequel le schéma doit se synchroniser est la vidange totale de la couche et donc sa disparition. Des résultats numériques confirmeront la bonne détection de l'événement de vidange de la couche par l'algorithme de synchronisation. En plus des gains en précision et stabilité apportés par la résolution implicite et synchronisée du couplage, ce problème permettra de mettre en valeur l'impact positif qu'a eu l'architecture de couplage sur la *qualité logicielle* de la plateforme. On verra en particulier que l'utilisation de tels schémas a permis la suppression de portions de codes présents dans les modèles de la plate-forme uniquement pour la correction de situations non physiques amenées par la résolution explicite du couplage.

Problème couplé issu d'une application industrielle PROCOR. Enfin, on s'intéressera à la résolution du couplage des modèles d'une vraie application industrielle PROCOR avec la nouvelle architecture de couplage. Ce cas pratique réel d'utilisation de l'architecture permettra de comprendre les différentes étapes du portage d'une application sous la nouvelle architecture et les possibilités offertes par les différentes fonctionnalités de l'architecture à la suite du portage. Des résultats numériques viendront montrer l'erreur commise par la résolution explicite du couplage avec les macro pas de temps de couplage utilisés pour les calculs industriels. On expliquera pourquoi de tels macro pas de temps sont imposés, d'une part par certains modèles physiques de l'application et, d'autre part, pour des raisons de performance et de coût des calculs. On détaillera le *schéma de couplage mixte explicite implicite* utilisé pour une résolution plus précise du couplage par rapport à la résolution explicite et permettant de respecter les contraintes sur le macro pas de temps. Des résultats numériques viendront confirmer les gains apportés par le nouveau schéma de couplage. Enfin, on détaillera les limites actuelles de l'application qui serviront de pistes pour d'éventuelles ouvertures à la suite de la thèse.

Tout au long des résultats numériques, on parlera parfois d'erreurs impor-

tantes créées par un schéma de couplage et des effets néfastes qu'elles peuvent avoir sur la simulation. Bien que vague et critiquable, ces termes sont souvent employés parce qu'aucune solution de référence n'est disponible pour évaluer qualitativement les erreurs. Tout ce que l'on peut évaluer dans de tels cas sont les déséquilibres aux interfaces entre modèles (couplage fort ou faible des modèles) et la synchronisation des modèles.

De plus, la *précision numérique* cible du schéma de couplage et de synchronisation (qui peut être très faible) n'est *absolument pas* à mettre en regard avec les incertitudes sur les paramètres d'entrées des modèles et la modélisation physique (qui peuvent être très importantes) qui sont inhérentes au contexte AG. C'est en résolvant le plus précisément possible le couplage des modèles que les connaissances globales des AG vont croître (une erreur importante sur la résolution du couplage pourrait masquer un phénomène physique important par exemple).

4.1 Problème couplé de conduction de la chaleur

Cette section a fait l'objet d'une publication dans "International Journal for Multiscale Computational Engineering" *et est retranscrite telle qu'elle.*

Journal for Multiscale Computational Engineering, x(x): 1–25 (2018)

SOLVING COUPLED PROBLEMS OF LUMPED PARAMETER MODELS IN A PLATFORM FOR SEVERE ACCIDENTS IN NUCLEAR REACTORS

L. Viot,[1,*] L. Saas,[1] & F. De Vuyst[2]

[1]*CEA, DEN, DTN/SMTA/LMAG, Cadarache, F-13108 Saint-Paul-lez-Durance, France*
[2]*LMAC EA 2222, UTC, Sorbonne Universités, 60200 Compiègne, France*

*Address all correspondence to: L. Viot, CEA, DEN, DTN/SMTA/LMAG, Cadarache, F-13108 Saint-Paul-lez-Durance, France, E-mail: louis.viot@cea.fr

This paper focuses on solving coupled problems of lumped parameter models. Such problems are of interest for the simulation of severe accidents in nuclear reactors at system level: these coarse-grained models allow for fast calculations for statistical analysis used for risk assessment and solutions of large problems when considering the whole severe accident scenario. However, this modeling approach has several numerical flaws. Besides, in this industrial context, computational efficiency is of great importance leading to various numerical constraints. The objective of this research is to analyze the applicability of explicit coupling strategies to solve such coupled problems and to design implicit coupling schemes allowing stable and accurate computations. The proposed schemes are theoretically analyzed and tested within CEA's PROCOR *platform on a problem of heat conduction and ablation representative of severe accidents in light water reactors. This problem is solved with coupled 0D lumped parameter models and coupled 1D models. Numerical results are discussed and allow us to emphasize the benefits of using the designed coupling schemes instead of the usual explicit coupling schemes.*

KEY WORDS: *Nuclear Engineering, Severe Accident, Multiphysics, Complex System, System-Level Simulation, Lumped Parameter Model, Coupling Scheme, Partitioned Approach, Explicit/Implicit Scheme, Stability Analysis, Accuracy, State Change Event, Computational Efficiency, Simulation Engine*

Acronyms. *LP, lumped parameter; ECS, explicit coupling scheme; ICS, implicit coupling scheme*

1. INTRODUCTION

Mathematical and numerical resolution of coupled multiscale and multiphysics problems is a substantial issue arising in several engineering fields. In particular in the field of severe accidents in nuclear reactors Sehgal (2012) Jacquemain (2013) coupling thermohydraulics, thermomechanics, thermochemistry, thermodynamics, neutronics phenomena with characteristic time, length and mass going from microseconds to days, millimeters to meters, kilograms to hundred of tons. In shuch a context, this requires the simulation of the whole accident scenario or just a part of it leading to coupled problems of different size. Furthermore, simulation can be used for statistical analysis, e.g. Monte-Carlo sensitivity analysis, involving a large number of calculations, or simply to have a finer and better understanding of some phenomenons. Altogether and because of the lack of full physical and phenomenological knowledge, a wide range of models for the underlying physical phenomena are used, e.g. stationary models, reduced models, mesh based models, etc. In particular, **lumped parameter models** (LP models), sometimes called "0D" models, simplify spatially distributed systems into discrete entities, e.g. partial differential equations become parameterized ordinary differential equations over time, in such a way that calculations require much less running time and a much lower computational cost. However, such simplifications come with a price to pay:

- LP models ignore the finite time of propagation of information of the continuous or space and time discretized

1543–1649/18/$35.00

model and instantly communicate and spread them;

- closure laws used in LP models are often described with highly non-linear functions of time and of the model internal state variables, e.g. correlations fitted from mesh based calculations or experimental results Bonnet and Seiler (1999) Zhang et al. (2015);
- state, phase or topological changes in the continuous time and space model, e.g. disappearance, vaporization, are turned into instant internal changes and sometimes discontinuous event triggered at a certain time by the LP model, which can be compared to event occurring in *Differential Algebraic Equations* (DAE) Mao and Petzold (2002). A state change event missed by the simulation engine can bring the coupled models in a non coherent and non physical state during a small window of time resulting in numerical errors, numerical instabilities which have to be avoided at all cost;

As a result, coupled problems of LP models can have fast and stiff transients which are numerically challenging to solve.

There are two approaches to solve such coupled problems : a **monolithic** approach which solves the governing equations describing each model simultaneously and a **partitioned** approach Felippa and Park (1980), its counterpart, in which coupled models are associated to the so-called *partitions* which are solved one at a time during the coupling iterations. One of the advantage of such partitioned approaches is that the solution of partitions can be done with different solvers adapted to the physical phenomenon involved in the partition. Moreover, great modularity and software reuse is achieved since partition solvers are assumed to be seen as "black box" by the partitioned problem with a set of inputs, a set of outputs and very limited internal details (e.g. derivative data) and thus can be easily exchanged. In this approach, partitions deliver physical quantities such as heat fluxes, forces, pressures, mass flow rates to other coupled partitions. In contrast with the monolithic approach, coupling equations between partitions are not part of a one block system of equations, instead, partitions are sharing data with external coupling equations corresponding to equilibrium conditions, e.g. heat flux equality between two thermal partitions sharing a geometrical interface Giles (1997) or temperature equality between a thermal partition and a thermodynamic partition sharing a common temperature. Because of possible decoupling effects between partitions, equilibrium conditions might not be enforced by the coupling algorithm leading to loosely or weakly coupled partitions. Therefore, there are two main classes of coupling schemes :

- **Explicit coupling schemes** (ECSs) Farhat et al. (1995, 1991, 2010); Farhat and Sobh (1990); Felippa and Park (1980); Guillard and Farhat (2000) which require only one call of each solver one after the other during each time step but which can only achieve weak coupling between the coupled partitions at the end of the time step.
- **Implicit coupling schemes** (ICSs) Degroote et al. (2009); Degroote and Vierendeels (2012); Ganine et al. (2013); Gerbeau and Vidrascu (2003); Joosten et al. (2009); Kassiotis et al. (2011); Küttler and Wall (2008); Michler et al. (2005); Minami and Yoshimura (2010); Vierendeels et al. (2007) which require several calls of solvers within an iterative loop until a certain convergence threshold. If the scheme has converged, equations and subdomains are strongly coupled at the end of the time step, up to a certain precision, and the monolithic scheme is recovered.

In this paper, we focus on the numerical solution of heterogeneous 0D and 1D models by partitioned approaches. In this very specific context of modeling, we show how ECSs are often not suitable for the solution of such problems and implicit treatment is often necessary. We furthermore explain how regular ICSs are modified in order to take into account state change events triggered by the models. By this way, the coupled models can be synchronized on potential events and discontinuities.

The paper is structured as follows. Section 2 presents phenomenology of severe accidents and the modelling of corium propagation during such accidents. Section 3 gives a brief overview of the lumped parameter mass and energy conservation governing equations in each subdomain and models used for the different previously cited phenomena ; for the sake of simplicity and conciseness, we only consider coupled thermalhydraulic phenomena. We believe that the extension to any other phenomena (e.g. thermochemistry, thermomechanic, neutronic, etc.) is similar. From there,

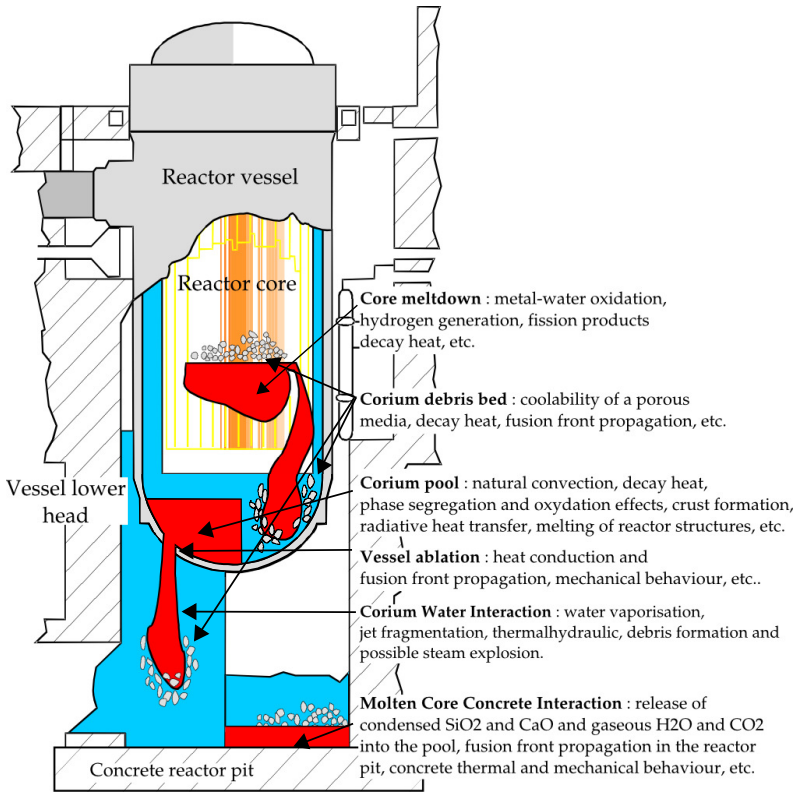


FIG. 1: Schematic representation of the corium propagation in a LWR during a SA and some of the several involved multiphysics and multiscale physical phenomena.

the coupled formulation is explained and the coupling algorithms are given in section section 4. In section 5, we give numerical analysis, computations and discussions of a coupled problem solved with various coupling schemes within the PROCOR platform Le Tellier et al. (2015). Finally, conclusion and opening remarks are given in section 6.

2. SEVERE ACCIDENTS PHENOMENOLOGY AND MODELLING

During a Severe Accident (SA) in a Nuclear Power Plant (NPP) in a Light Water Reactor (LWR), the lack of fuel cooling (due to a break in the primary circuit for example) in the reactor vessel leads to core heat up and melt down forming the so-called corium pool (oxidic and metallic liquid materials). Then, as described in Fig. 1, the melt propagates from the core to the lower plenum of the Reactor Pressurized Vessel (RPV). The volumetric heat generation associated with decay heat induces natural convection in the pool that, combined with phase segregation effects, determines the heat flux distribution at the pool interface and, in particular, the thermal load imposed to the lower plenum of the vessel. In case of vessel rupture (because of insufficient external cooling of the vessel for example), the melt flows down to the concrete reactor pit, the so-called Molten Core Concrete Interaction (MCCI), and possibly, in case of concrete failure, to the outside environment leading to a large release of low-volatile radioactive products to the reactor surroundings. Consequently, the corium propagation involves several coupled physics (e.g. neutronic in the core meltdown, thermohydraulic for the natural convection in the corium pool, thermochemistry for phase segregation effects in the corium pool, thermomechanic for ablation kinetic and possible rupture of the reactor vessel, etc.) located at different areas (e.g. in the reactor core, in the lower plenum of the vessel, in the concrete reactor pit, etc.), as described in Fig. 1.

In the context of SA codes for LWRs, because of the complexity of the many coupled phenomena at play and the

possible lack of physical data, phenomenological and modelling knowledge, *a sensitivity/uncertainty-driven approach* has been established. This approach supplements integral codes based on bounding assumptions, e.g. steady state situations, which usually do not take into account the possible important impact of transient phenomena, e.g. the formation of a "thin" steel layer on top of the corium pool in the lower plenum of the reactor vessel associated to heat fluxes concentration at the vessel, the so-called focusing effect, leading to possible reactor vessel failure Rempe et al. (1998); Theofanous et al. (1997). In this approach, phenomenological analyses based on available data, on physical models and knowledge about the corium behavior are carried out for different possible corium propagation scenarios for given reactor and SA management strategy. Then, these analyses allow us to identify the first-order phenomena, the most plausible corium propagation sequences and are then used as guidelines for the model assembly and parametrization in a dedicated transient code, e.g. the CEA `PROCOR` SAs code for LWRs Le Tellier et al. (2015) used in this work. The statistical studies that can be performed with such codes are used both as an assessment of specific SA design features or modifications in a reactor and for prioritizing modelling issues and associated R&D.

Consequently, the phenomenological and sensitivity/uncertainty-driven approaches impose some modelling constraints on the coupled problem of the corium propagation :

- for statistical analyses, e.g. Monte-Carlo sensitivity analyses involving a large number of calculations, models should be effective in terms of running time and a computational cost, e.g. lumped parameter models (LP).
- the coupling between different physics and the effective interaction betweens subdomains are dependent on the reactor and the SA scenario and management strategy. Furthermore, the graph of couplings might evolve with model appearances or disappearances, e.g. vessel rupture or water evaporation.
- statistical analyses allow us to identify important modelling issues and to prioritize modelling efforts in such a way that, during the industrial life cycle of the coupled problem, models are evolving and might have to change very often.

The partitioned approach Felippa and Park (1980) is particularly well adapted to these constraints since it allows us to use adapted solver for each of the multiscale and multiphysics phenomena of the coupled problem and offers great code reuse and modularity. In this partitioned approach, the coupled problem associated with the corium propagation takes the form of a complex system Boccara (2010). To represent such systems, a general coupling formalism has been created in Sec. 4.

Furthermore, the coupled problem of corium propagation can take different forms :

- different coupled physics located at the same subdomain, e.g. in the corium pool in the lower plenum of the vessel the coupling between natural convection phenomena (induced by volumetric heat generation associated with decay heat) and phase segregation phenomena (leading to separated oxidic and metallic liquid layers).
- different physics located at different coupled subdomains.

In this work, we focus on the second point described in Sec. 3 while the governing equations in each subdomain are presented in Sec. 3.1. The considered couplings between subdomains represent the heating (by conduction heat transfer) and ablation (associated with a Stefan problem Le Tellier et al. (2017)) from one hot subdomain to the heated and ablated subdomain. They are described in Sec. 3.2. This problem is of prime interest for SA in LWRs when evaluating :

- the melting of cladding materials associated with metal-water oxidation and hydrogen generation Haurais (2016).
- the possible formation of a crust surrounding the corium pool inside the lower plenum of the vessel Carénini et al. (2018).
- the possible vessel rupture due to its heating and ablation by the imposed thermal load from the corium pool which is of prime interest when evaluating the change of success of the In-Vessel-Retention (IVR) management strategy Theofanous et al. (1997) by external cooling of the vessel.

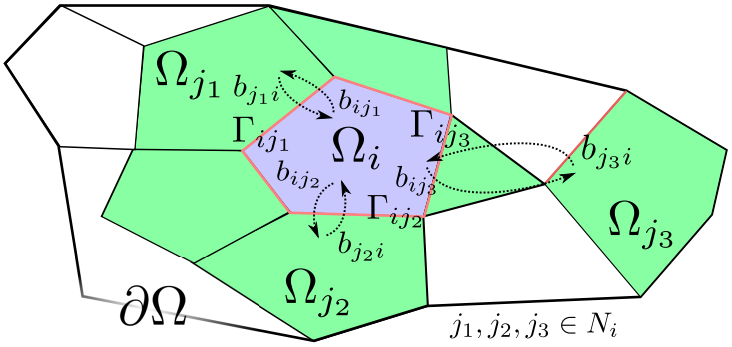


FIG. 2: The abstract representation of domain decomposition of Ω focused on domain Ω_i and its neighborhood $\Omega_{j\in N_i}$.

- the ablation of the concrete reactor pit in case of rupture of the pressurized reactor vessel leading to corium flowing down in the reactor pit, the so-called Molten Core Concrete Interaction (MCCI) Spindler et al. (2006).

Models of the coupled problem used in this work are extracted from the PROCOR software platform in such a way that they are representative of industrial calculations. Furthermore, while the coupled problem remains simple in comparison to complex systems arising during SAs in LWRs, they allow us to reproduce the numerical pathologies in terms of coupling and synchronization associated with ECSs which are taken care of with ICSs.

3. GOVERNING EQUATIONS

To simplify, we assume that the coupled problem to solve consists of different physics located at different subdomains. Figure 2 depicts a non overlapping domain decomposition of domain $\Omega = \cup_{i\in[\![1,m]\!]}\Omega_i$. Neighboring domains Ω_j coupled with domain Ω_i communicating through interface Γ_{ij} can be actual geometrical neighbor of domain Ω_i, i.e. $\Gamma_{ij} = \partial\Omega_i \cap \partial\Omega_j \neq \emptyset$ or can be distant neighbor of domain Ω_i with whom it has linked but distant phenomena, i.e. $\partial\Omega_i \cap \partial\Omega_j = \emptyset$ and Γ_{ij} is a part of frontier $\partial\Omega_i$ of domain Ω_i. Neighbors of domain Ω_i are represented by the set $N_i = \{j \ / \ \exists\ \Gamma_{ij}\}$, with cardinality denoted by $n_i = \text{card}(N_i)$. Finally, frontier of domain Ω_i can be calculated by $\partial\Omega_i = \Gamma_i \cup (\cup_{j\in N_i}\Gamma_{ij})$. Subdomain Ω_i is described in terms of mass denoted by m_i [kg] and average temperature denoted by T_i [K]. We have :

$$m_i = \int_{\Omega_i} \rho \, \mathrm{d}V, \quad T_i = \frac{1}{V_{\Omega_i}} \int_{\Omega_i} T \, \mathrm{d}V.$$

The vector of state variables of subdomain Ω_i is denoted by $\mathbf{u}_i = (m_i, T_i)^{\mathrm{t}}$. Note that in this article we decide to represent each subdomain in term of average temperature even though they could be represented in term of average enthalpy.

Interface variables are heat fluxes ϕ_{ij} [W.m^{-2}], temperature T_{ij} [K], mass flow rate $\dot{m}_{ij}$ [kg.s^{-1}] or surface area S_{ij} [m^2]. These variables are gathered in vector $\mathbf{b} = \{(\phi_{ij}, T_{ij}, \dot{m}_{ij}, S_{ij})^{\mathrm{t}}\}$ for $i \in [1, m]$ and $j \in N_i$. The interface projector $\mathcal{P}_{ij}$ allows us to get interface Γ_{ij} vector variables $\mathbf{b}_{ij} = (\phi_{ij}, T_{ij}, \dot{m}_{ij}, S_{ij})^{\mathrm{t}}$ from vector $\mathbf{b}$ with $\mathcal{P}_{ij}\mathbf{b} = \mathbf{b}_{ij}$.

We first describe the lumped parameter governing equations for each subdomain in section 3.1 and then present the interface equations between subdomains in section 3.2.

3.1 Subdomain lumped parameter equations

Subdomain Ω_i equations are expressed in terms of mass and energy macroscopic conservation equations (momentum conservation equations are modeled by closure laws, e.g. correlations for heat transfer coefficient). They are obtained from the local conservation equations, here Navier-Stokes equations under the Boussinesq approximation for liquid domains and heat equations for solid domains, integrated over the corresponding subdomain (see Le Tellier et al.

(2017)). Tightly linked to the local physical model, this approach leads to the so-called LP model or "0D" model of the subdomain described by the two ordinary differential equations (ODE)

$$\frac{\mathrm{d}m_i}{\mathrm{d}t} = \sum_{j\in N_i} \dot{m}_{ij} \quad \text{in } \Omega_i, \tag{1}$$

$$m_i C_{p_i} \frac{\mathrm{d}T_i}{\mathrm{d}t} + \sum_{j\in N_i} \dot{m}_{ij} C_{p_i}(T_i - T_{ij}) = \sigma_i \phi_i S_i + \sum_{j\in N_i} \sigma_{ij}\phi_{ij} S_{ij} + m_i \dot{q}_i \quad \text{in } \Omega_i \tag{2}$$

with m_i [kg] the mass and T_i [K] the average temperature of Ω_i, ϕ_i [W.m^{-2}] the heating ($\sigma_i = 1$) or cooling ($\sigma_i = -1$) heat flux through boundary $\Gamma_i = \partial\Omega \cap \partial\Omega_i$ with temperature T_{b_i} and area S_i [m^2], ϕ_{ij} the heating or cooling heat flux and $\dot{m}_{ij}$ [kg.s^{-1}] the algebraic mass flow rate through Γ_{ij} with temperature T_{ij} and area S_{ij}. Finally, C_{p_i} [J.kg^{-1}.K^{-1}] is the heat capacity and $\dot{q}_i$ [W.kg^{-1}] is the residual power per mass unit coming from fission products of subdomain Ω_i.

The previous physical parameters are obtained from closure laws described hereafter in section 3.2. In particular, geometry dependent values, i.e. the characteristic length of a domain or the surface of an interface or the volume of a domain, are given by algebraic geometry equations. They take the form of algebraic functions, e.g. for the previous surface :

$$S_{ij} = S_{ij}(\rho_i, e_i, V_i) \tag{3}$$

with e_i [m] the characteristic length and V_i [m^3] the volume of domain Ω_i. Altogether, ordinary differential Eqs. 1 and 2 combined with expressions like Eq. 3 can be considered as *Differential Algebraic Equations* (DAE's) Petzold (1982) describing the LP model.

3.2 Interface equations

In the present article, interfaces between two subdomains Ω_i and Ω_j can be of two types : free moving or fixed boundaries. Physically moving boundaries are, for example, boundaries between a liquid and a solid domain exchanging melted or solid materials. They are associated with a plane fusion solidification front corresponding to the Stefan condition at the interface (see Le Tellier et al. (2017) for further details). Fixed interfaces correspond to thermal equilibrium assuming no mass exchange through the interface and thermal conduction.

In case of a mobile interface, equilibrium conditions at the interface Γ_{ij} are given by

$$\dot{m}_{ij} = -\dot{m}_{ji} \quad \text{in } \Gamma_{ij}, \tag{4}$$

$$\phi_{ij} S_{ij} = -\phi_{ji} S_{ji} + \Delta\mathcal{H}^{\text{fus.}} \dot{m}_{ij} \quad \text{in } \Gamma_{ij}, \tag{5}$$

$$T_{ij} = T_{ji} = T^{\text{fus.}} \quad \text{in } \Gamma_{ij} \tag{6}$$

with $\Delta\mathcal{H}^{\text{fus.}}$ the fusion enthalpy and $T^{\text{fus.}}$ the fusion temperature of domain Ω_i, both assumed to be fixed. In particular, those conditions stipulate that the mass flow rate should be the same on both sides of the interface for mass conservation and that the heat fluxes should respect the Stefan condition. To simplify subdomain materials are treated as pure body and no thermo-chemistry is considered.

In case of a fixed interface, the thermal equilibrium conditions at interface Γ_{ij} are given by

$$\dot{m}_{ij} = -\dot{m}_{ji} = 0 \quad \text{in } \Gamma_{ij}, \tag{7}$$

$$T_{ij} = T_{ji} \quad \text{in } \Gamma_{ij}, \tag{8}$$

$$\phi_{ij} S_{ij} = -\phi_{ji} S_{ji} \quad \text{in } \Gamma_{ij} \tag{9}$$

Obviously, with the previous equations, appropriate closure laws for interface heat fluxes and temperatures are required. In traditional mesh based models, interface variables are given by a projection operator which are coherent with the domain equations, e.g. the restriction of the variables values over all the domain Ω_i to its interface $\Gamma_{ij} =$

$\partial\Omega_i \cap \partial\Omega_j$, and thus are consistent with the physical equations. For LP models, interface variable $\mathbf{b}_{ij}$ is calculated from spatially averaged data $\mathbf{u}_i$ from domain Ω_i and data $(\{\mathbf{b}_{ij}\}_{j\in N_i}) \overset{def}{=} \mathbf{b}_{i\star}$ from the interfaces. They are given by closure laws which take the form of algebraic functions $\mathbf{b}_{ij} = \mathbf{b}_{ij}(\mathbf{u}_i, \mathbf{b}_{i\star})$. Thus they propagate data instantly, from one specific interface to all the other ones.

If the domain Ω_i is solid, such closure law functions for heat fluxes $\phi_{i\bullet}$ can be calculated from the heat diffusion conduction equation under certain assumptions and approximations. A comparison of those different approximate models with a reference solution given by a finite element discretization of the heat conduction equation can be found in Le Tellier et al. (2017). For example, the *stationary* model uses a 1D cylindrical with adiabatic lateral boundaries approximation and assumes a quadratic temperature profile in the solid domain. Conduction heat fluxes at interfaces Γ_{ij} and Γ_{ik} associated with the upper and lower cylinder surfaces are then given by the temperature derivative w.r.t the spatial direction leading to the closure law functions Le Tellier et al. (2017) :

$$\phi_{ij} = \phi_{ij}(\mathbf{u}_i, \mathbf{b}_{ik}) = \lambda_i \frac{6T_i - 4T_{ij} - 2T_{ik}}{e_i}, \tag{10}$$

$$\phi_{ik} = \phi_{ik}(\mathbf{u}_i, \mathbf{b}_{ij}) = \lambda_i \frac{6T_i - 4T_{ik} - 2T_{ij}}{e_i}. \tag{11}$$

Note, for instance, the instant propagation of interface Γ_{ik} temperature T_{ik} to interface Γ_{ij} in Eq. 10 indicating that the *stationary* model gives closure law functions which propagate boundary related data **instantly** between interfaces of the domain.

Let us mention that, in the literature, one can find other nonlinear closure laws like e.g.:

- $\phi_{ij}(\mathbf{u}_i, \mathbf{b}_{i\star}) \propto e_i^{\alpha}(T_i - T_{ij})^{\beta}$ with $\alpha, \beta \in \mathbb{R}$ for convective heat transfer, e.g. used to calculate the heat fluxes from the corium pool in the lower plenum of the vessel to the vessel wall leading to its ablation and possible rupture.
- $\phi_{ij}(\mathbf{u}_i, \mathbf{b}_{i\star}) \propto {T_{ij}}^4$ for radiative heat transfer, e.g. used to calculate the heat fluxes from the top of the corium pool in the lower plenum of the vessel to the upper reactor structures or part of the vessel wall not in direct contact with the pool.

For the sake of simplicity, in this paper we will only consider heat transfer by conduction. Consequently, the closure laws for heat fluxes in the macroscopic energy conservation equation Eq. 2 will be given by the LP Eqs. 10 and 11*. The modelling of more physics is possible and is given as perspectives in Sec. 6 but would limit the numerical analyses in Sec.5. However, the considered coupled problem is still highly relevant for SAs in LWRs when considering, for instance, the ablation of the reactor pressurized vessel and its possible rupture in the frame of the In-Vessel-Retention accident management strategy Theofanous et al. (1997). Indeed in that case, the liquid and solid subdomains are thins and the liquidus and solidus temperature of the steel of the vessel are very close in such a way that the pure body and very thin interface assumptions done with the considered coupled problem can be made.

4. COUPLING FORMULATION AND SCHEMES

4.1 Discretized coupling formulation and coupled problem

LP conservation Eqs. 1 and 2 are then discretized in time at a lower level with a method of choice (e.g. explicit, implicit, Euler, Runge-Kutta, multistep methods, etc.) and then coupled in time on a higher level resulting in a two-level time scheme. At the highest level, those discretized equations are solved and coupled between times $t^0, t^1, \ldots, t^n, \ldots$ with a macro time step Δt and synchronized at each of these times. At the lowest level, each subdomain manages its own time integration scheme, its own micro time step δt and its own internal time. The integration scheme used by the subdomain is assumed to be adapted to the physical local problem. In the following, discretized values evaluated at time t^n are denoted by the superscript n.

*Note that for comparison, these heat fluxes will also be calculated by 1D heat conduction models in Sec. 5.

We adopt the following notations. For each coupled subdomain $\{\Omega_i\}_{i\in[1,m]}$, discretized equations are represented by a function $\mathcal{F}_i^{\Delta t}$ used to solve and advance in time the problem of one time step Δt. This function takes as parameters the state vector $\mathbf{u}_i$ and the input interface variables $\{\mathbf{b}_{ji}\}_{j\in N_i} \overset{\text{def.}}{=} \mathbf{b}_{\star i}$ of domain Ω_i. Equations of the coupled problem are then given by

$$\begin{cases} \mathcal{F}_1^{\Delta t}(\mathbf{u}_1, \{\mathbf{b}_{j1}(\mathbf{u}_j,\ \mathbf{b}_{\star j})\}_{j\in N_1}) = 0 \\ \mathcal{F}_2^{\Delta t}(\mathbf{u}_2, \{\mathbf{b}_{j2}(\mathbf{u}_j,\ \mathbf{b}_{\star j})\}_{j\in N_2}) = 0 \\ \qquad\vdots \\ \mathcal{F}_m^{\Delta t}(\mathbf{u}_\mathbf{m}, \{\mathbf{b}_{jm}(\mathbf{u}_j,\ \mathbf{b}_{\star j})\}_{j\in N_m}) = 0 \end{cases} \tag{12}$$

in such a way that the inter dependencies between domain Ω_i and its neighbors are highlighted by the closure law functions $\{\mathbf{b}_{ji}\}_{j\in N_i}$ which is function of the coupled subdomain internal variable $\mathbf{u}_j$ and interface variables $\mathbf{b}_{\star j}$, which might in turn need values from domain Ω_i to be calculated.

Those interface dependencies highlight the input-output relationship between interface variables of the different subdomains. This suggests that solvers can be seen as closed entities or "black-boxes" taking input interface variables $\mathbf{b}_{\star i}$ from neighboring subdomains and giving back output interface variables $\mathbf{b}_{i\star}$ to neighboring subdomains. Therefore, it seems natural to represent solver of domain Ω_i as a function $\mathcal{M}_i^{\Delta t} : \mathbb{R}^{3\times n_i} \mapsto \mathbb{R}^{3\times n_i}$ in which we explicitly hide subdomain state variable $\mathbf{u}_i$ and its integration, and only the input and output interface variables are visible. It is possible that we have access to limited information about them, e.g. no derivative data. For instance, solver of domain Ω_i is defined by

$$\mathbf{b}_{i\star} = \mathcal{M}_i^{\Delta t}(\mathbf{b}_{\star i}). \tag{13}$$

From Eq. 13, we can define the coupled problem at interface Γ_{ij} in term of solvers :

$$\mathbf{b}_{ij} = \mathcal{P}_{ij} \circ \mathcal{M}_i^{\Delta t}\left(\mathbf{b}_{ji}, \{\mathbf{b}_{ki}\}_{k\in N_i,\ k\neq j}\right), \tag{14}$$

$$\mathbf{b}_{ji} = \mathcal{P}_{ji} \circ \mathcal{M}_j^{\Delta t}\left(\mathbf{b}_{ij}, \{\mathbf{b}_{kj}\}_{k\in N_j,\ k\neq i}\right) \tag{15}$$

with $\mathcal{P}_{ij}$ the interface Γ_{ij} projector. Those equations emphasize the "action-reaction" at interface between the two domains : a slight modification of interface variable $\mathbf{b}_{ji}$ will in turn change the associated interface variable $\mathbf{b}_{ij}$, and vice versa. The strength of the associated coupling is represented by the Jacobian matrices $\{\frac{\partial \mathcal{M}_i^{\Delta t}}{\partial \mathbf{b}_{ki}}\}_{k\in N_i}$ and $\{\frac{\partial \mathcal{M}_j^{\Delta t}}{\partial \mathbf{b}_{kj}}\}_{k\in N_j}$. Moreover it is difficult to evaluate this strength since they may not be available.

Combining Eq. 14 with Eq. 15 gives the widely used fixed point equation at interface Γ_{ij} (see Mehl et al. (2016))

$$\mathbf{b}_{ij} = \mathcal{P}_{ij} \circ \mathcal{M}_i^{\Delta t}\left(\mathcal{P}_{ji} \circ \mathcal{M}_j^{\Delta t}\left(\mathbf{b}_{ij}, \{\mathbf{b}_{kj}\}_{k\in N_j,\ k\neq i}\right), \{\mathbf{b}_{ki}\}_{k\in N_i,\ k\neq j}\right) \tag{16}$$

which allows us to define the residual operator $\mathcal{R}_{ij}$ at interface Γ_{ij} by

$$\mathcal{R}_{ij}(\mathbf{b}_{ij}) \overset{def}{=} \mathcal{P}_{ij} \circ \mathcal{M}_i^{\Delta t}\left(\mathcal{P}_{ji} \circ \mathcal{M}_j^{\Delta t}\left(\mathbf{b}_{ij}, \{\mathbf{b}_{kj}\}_{k\in N_j,\ k\neq i}\right), \{\mathbf{b}_{ki}\}_{k\in N_i,\ k\neq j}\right) - \mathbf{b}_{ij} \tag{17}$$

for a guess candidate $\mathbf{b}_{ij}$. The residual $\mathcal{R}_{ij}(\mathbf{b}_{ij})$ will express the imbalance created at interface Γ_{ij} : if domains Ω_i and Ω_j are strongly coupled, equilibrium conditions at interface are fulfilled and the interface residual is null, otherwise the two domains are only weakly coupled.

4.2 Coupling schemes

4.2.0.1 Explicit coupling schemes (ECSs).

Referred to as "conventional serial staggered" in Felippa and Park (1980), ECSs solve the coupled problem with one call of solver per time step. They are based on a Gauss-Seidel semi-explicit solution of the coupled problem Eq. 12.

Obviously, another scheme based on a fully-explicit or Jacobi solution can be used to allow for more algorithm parallelism. If we assume that solver $\mathcal{M}_i^{\Delta t}$ is solved before solver $\mathcal{M}_j^{\Delta t}$, the scheme solves at interface Γ_{ij} :

$$\mathbf{b}_{ij}^{n+1} = \mathcal{P}_{ij} \circ \mathcal{M}_i^{\Delta t} \left(\mathbf{b}_{ji}^{n}, \{\mathbf{b}_{ki}^{\bullet}\}_{k \in N_i,\ k \neq j} \right), \tag{18}$$

$$\mathbf{b}_{ji}^{n+1} = \mathcal{P}_{ji} \circ \mathcal{M}_j^{\Delta t} \left(\mathbf{b}_{ij}^{n+1}, \{\mathbf{b}_{kj}^{\bullet}\}_{k \in N_j,\ k \neq i} \right) \tag{19}$$

with $\mathbf{b}_{ki}^{\bullet}$ is evaluated at t^n or t^{n+1} if solver $\mathcal{M}_k^{\Delta t}$ is called before or after solver $\mathcal{M}_i^{\Delta t}$.

While potentially attractive and fast since only one call per solver is done during each time step, it is well known that this scheme yields poor accuracy and stability issues Piperno et al. (1995). Because of the *time-lag* caused by the semi-explicit resolution, it is unlikely that interface residuals defined by Eq. 17 are null and equilibrium conditions are not enforced at interface. Besides, despite several improvements and studies in Farhat et al. (1995) Guillard and Farhat (2000) Farhat and Sobh (1990) Farhat et al. (2010) Farhat et al. (1991) Piperno and Farhat (2001), the weak coupling reached by explicit coupling schemes is often not enough and only a strong coupling at the end of the time step can ensure proper stability properties Causin et al. (2005) Giles (1997).

4.2.0.2 Implicit coupling schemes (ICSs).

A way to fix the *time-lag* is to use implicit coupling between subdomains. At interface Γ_{ij}, implicit coupling gives :

$$\mathbf{b}_{ij}^{n+1} = \mathcal{P}_{ij} \circ \mathcal{M}_i^{\Delta t} \left(\mathbf{b}_{ji}^{n+1}, \{\mathbf{b}_{ki}^{n+1}\}_{k \in N_i,\ k \neq j} \right), \tag{20}$$

$$\mathbf{b}_{ji}^{n+1} = \mathcal{P}_{ji} \circ \mathcal{M}_j^{\Delta t} \left(\mathbf{b}_{ij}^{n+1}, \{\mathbf{b}_{kj}^{n+1}\}_{k \in N_j,\ k \neq i} \right) \tag{21}$$

We clearly see that Eqs. 20 and 21 give discrete interface values respecting the fixed point Eq. 16 leading to a null interface residual. Hence, interface Γ_{ij} is at equilibrium and a strong coupling between equations is obtained. However, while being mathematically interesting, Eqs. 20 and 21 do not allow to decouple the two equations at each time step and it is often more convenient to use iterative methods to solve them. By iterative methods we mean methods using iterative processes inside one coupling iteration, i.e. between times t^n and t^{n+1}, until a convergence criterion is reached. In that case interface variables *verify* (with an error bounded by the convergence criterion) at each interface the associated fixed point Eq. 16 leading to a strong coupling between equations of the coupled problem. At interface Γ_{ij} between domain Ω_i and Ω_j, the block Gauss-Seidel coupling scheme, also known as staggered or partitioned coupling scheme, is an iterative method based on Newton-Raphson iterations of the fixed point Eq. 16 given by

$$\mathbf{b}_{ji}^{n+1,k+1} = \mathbf{b}_{ji}^{n+1,k} + \left(\frac{\mathrm{d}\mathcal{R}_{ji}}{\mathrm{d}\mathbf{b}_{ji}} \Big|_{\mathbf{b}_{ji}^{n+1,k}} \right)^{-1} (-\mathcal{R}_{ji}(\mathbf{b}_{ji}^{n+1,k})) \tag{22}$$

with the residual $\mathcal{R}_{ji}(\mathbf{b}_{ji}^{n+1,k})$ at iteration k defined by

$$\mathcal{R}_{ji}(\mathbf{b}_{ji}^{n+1,k}) = \mathbf{r}_{ji}^{n+1,k} = \mathcal{M}_j^{\Delta t} \circ \mathcal{M}_i^{\Delta t}(\mathbf{b}_{ji}^{n+1,k}) - \mathbf{b}_{ji}^{n+1,k} \tag{23}$$

which allows us to define a convergence criterion for the iterative process by

$$\frac{||\mathcal{M}_j^{\Delta t} \circ \mathcal{M}_i^{\Delta t}(\mathbf{b}_{ji}^{n+1,k}) - \mathbf{b}_{ji}^{n+1,k}||}{||\mathbf{b}_{ji}^{n+1,k}||} = \frac{||\mathbf{r}_{ji}^{n+1,k}||}{||\mathbf{b}_{ji}^{n+1,k}||} \leq \epsilon_{\text{rel}} \tag{24}$$

with ϵ_{rel} the relative stopping criterion for the subiterative process. *Gauss-Seidel* or *staggered* symbolizes the way interfaces data are sequenced through solvers (or *block*) inside coupling iterations.

4.2.0.3 Remark on Jacobian matrices and associated solvers.

In Eq. 22, the Jacobian matrix or its inverse are usually not known and/or not calculable since solvers are seen as black-box solvers, thus the iterations can only be approximated by approximation of the Jacobian matrix or its inverse leading to Quasi-Newton techniques. In Gerbeau and Vidrascu (2003), reduced order models are used to calculate the Jacobian matrix. In Michler et al. (2005), Newton-Krylov subiterations are used to approximate the Jacobian matrix leading to the so-called *Jacobian free Newton-Krylov* method. From previous residuals, Degroote et al. (2009) solve least-squares problems to approximate the inverse of the Jacobian matrix leading to *Interface Quasi-Newton - Inverse Least-Squares* (IQN-ILS) methods while Vierendeels et al. (2007) approximate the Jacobian matrix with similar methods leading to *Interface Quasi-Newton - Least-Squares* (IQN-LS). Multigrid Quasi-Newton methods are also studied in Degroote and Vierendeels (2012). A recent review of different Quasi-Newton methods can be found in Minami and Yoshimura (2010) for fluid-structure interaction and in Ganine et al. (2013) for thermal-structure interaction.

Much cheaper methods use relaxation techniques for the interface variable iterations which consists in using an approximation of the inverse of the interface residual Jacobian matrix of the form

$$\left(\frac{\mathrm{d}\mathcal{R}_{ji}}{\mathrm{d}\mathbf{b}_{ji}}\Big|_{\mathbf{b}_{ji}^{n+1,k}}\right)^{-1} \approx -\omega^k\, I \tag{25}$$

for which the fixed point iterations with dynamic relaxation are given by

$$\mathbf{b}_{ji}^{n+1,k+1} = \omega^k \mathcal{M}_j^{\Delta t} \circ \mathcal{M}_i^{\Delta t}(\mathbf{b}_{ji}^{n+1,k}) + (1-\omega^k)\mathbf{b}_{ji}^{n+1,k}. \tag{26}$$

Note that for $\omega^k = 1$, iterations given by Eq. 26 are classical Picard iterations which converge generally only linearly and very slowly. In Küttler and Wall (2008), the authors show that the block Gauss-Seidel iterative method with relaxation techniques for the resolution of equation Eq. 22 can be very efficient at a surprisingly low cost in comparison to more elaborated Quasi-Newton methods. Besides, it is shown in Ramière and Helfer (2015) that these techniques are also proven to be very competitive in comparison to the direct solution of the nonlinear problem given by Eq. 22 with traditional Newton-Raphson methods using derivative data. In this paper, we focus on block Gauss-Seidel iterative method with relaxation techniques.

For instance, the solution of equation Eq. 22 by the Steffensen's method accelerated with Aitken's delta-squared Δ^2 method Aitken (1927) can be cast into fixed point iterations with dynamic relaxation leading to methods of order 2, see Ramière and Helfer (2015). Less costly but with a convergence rate of the golden ratio $\frac{1+\sqrt{5}}{2}$, the fixed point iterations given by the secant method can also be seen as fixed point iterations with dynamic relaxation leading to

$$\omega^k = -\omega^{k-1}\frac{\langle\, \mathbf{r}_{ji}^{n+1,k} - \mathbf{r}_{ji}^{n+1,k-1},\, \mathbf{r}_{ji}^{n+1,k-1}\rangle}{\langle\, \mathbf{r}_{ji}^{n+1,k} - \mathbf{r}_{ji}^{n+1,k-1},\, \mathbf{r}_{ji}^{n+1,k} - \mathbf{r}_{ji}^{n+1,k-1}\rangle}. \tag{27}$$

One can also use constant relaxation given by a constant parameter ω. However, this requires the determination of the best relaxation parameters, i.e. leading to the highest rates of convergence, which is highly problem-dependent.

4.2.0.4 Implicit coupling schemes for event detection and model synchronization.

In LP modeling, state transitions are triggered by internal events corresponding to activation of threshold functions. This activation is dependent on input from the coupled models and thus times of events are unknowns of the coupled problem. As depicted in Fig. 3, each state has its own solver, its own set of equations and its own interface and boundary conditions.

State transition can lead to discontinuities of state or interface variables. A missed event can lead to inconsistent and incoherent physical states, e.g. the disappearance of a model not seen by other coupled models, and/or mathematical difficulties with non-smooth functions in the coupled problem Eq. 12 bringing usual theorems out of their scope of validity, with possibly breakdown issue like divergence of Newton iterations.

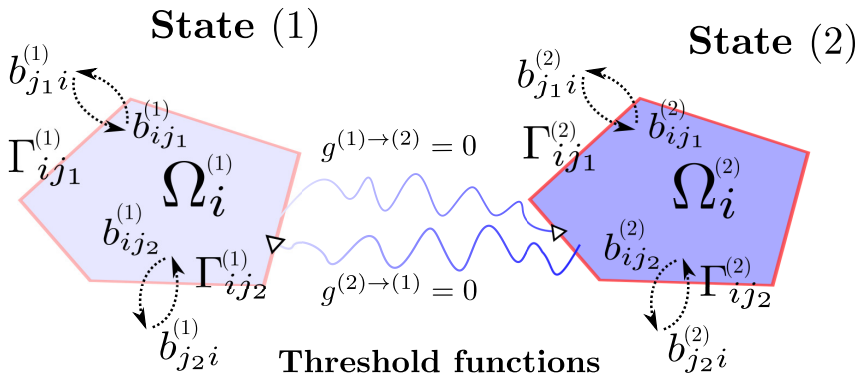


FIG. 3: The abstract representation of states $\Omega_i^{(1)}$ and $\Omega_i^{(2)}$ of domain Ω_i. Each state has its own interfaces $\Gamma_\bullet^{(1)}$ and $\Gamma_\bullet^{(2)}$ with associated variables $\mathbf{b}_\bullet^{(1)}$ and $\mathbf{b}_\bullet^{(2)}$. A state transition occurs when a transition function crosses zero.

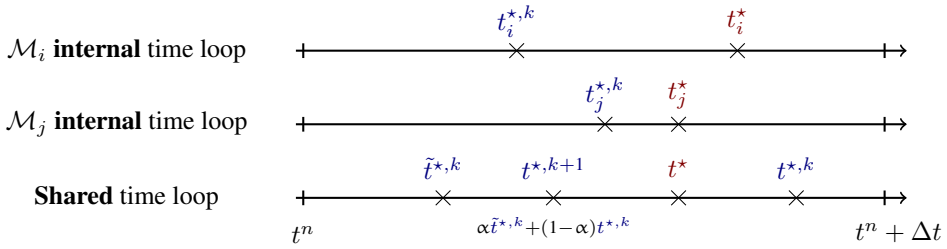


FIG. 4: Shared coupling time loop and internal time loops of solvers $\mathcal{M}_i$ and $\mathcal{M}_j$. Internal events at times $t_i^\star$ and $t_j^\star$ are to be detected by the scheme in order to synchronize on time $t^\star$ of the first event. An ICS detects event at times $t_i^{\star,k}$ and $t_j^{\star,k}$ and has to calculate the new iteration $t^{\star,k+1}$ in such a way that $t^{\star,\infty} \approx t^\star$.

During one coupling time step Δt, the coupling scheme must be able to adapt its time step to synchronize all the models on the time of the first event. However, synchronization strategies are hard to achieve with ECSs which can often only synchronized coupled models at each coupled time step leading to time detection errors of order $\mathcal{O}(\Delta t)$. Hereafter, we describe how the previous ICSs can be used for event detection to ensure proper synchronization between models.

Let us consider the coupling at interface Γ_{ij} between domain i and j. As depicted in Fig. 4, internal events are triggered during one coupling time step Δt by solver $\mathcal{M}_i$ and/or solver $\mathcal{M}_j$ at respective times $t_i^\star$ and $t_j^\star$ between t^n and $t^n + \Delta t$. When an event occurs, the solver stops its computation and do not compute the whole coupling time step Δt. Solvers should be synchronized on the first triggered event at time $t^\star = t_j^\star$.

During a fixed point iteration of an ICS starting at time t^n and ending at the previously calculated time of first event $t^{\star,k}$, events are trigerred at times $t_i^{\star,k}$ and $t_j^{\star,k}$ by solvers. The aim is to build an iterative algorithm calculating the new iteration $t^{\star,k+1}$ in such a way that : $t^{\star,\infty} \approx t^\star$. Alg. 1 gives a sequence to calculate this new iteration. At convergence, the coupled problem has been solved between times t^n and $t^{n+1} = t^{\star,\infty}$. The next coupling time step then starts at time t^{n+1} with the same algorithm.

4.2.0.5 Concluding remarks regarding the designed iterative algorithm.

At convergence $k = \infty$ of a coupling time step between times t^n and t^{n+1} at interface Γ_{ij}, we get that :

- Model i and model j are strongly coupled because the fixed point Eq. 16 is verified with at most an error

Algorithm 1: Calculate the new iteration $t^{\star,k+1}$

Compute a new fixed point iteration between solvers $\mathcal{M}_i^{\Delta t}$ and $\mathcal{M}_j^{\Delta t}$ with an ICS;
if *an event has occurred for model* $l \in \{i, j\}$ **then**
    $t_l^{\star,k} \leftarrow$ the corresponding event time;
else
    $t_l^{\star,k} \leftarrow t^n + \Delta t$;
    /* the whole coupling time step Δt was computed. */
end if
$\tilde{t}^{\star,k} \leftarrow \min\left(t_i^{\star,k}, t_j^{\star,k}\right)$;
if $(|\tilde{t}^{\star,k} - t^{\star,k}|/\Delta t < \epsilon_{\text{rel}})$ **and** (*the ICS has converged at interface* Γ_{ij}) **then** /* ϵ_{rel} being a tolerance given by the ICS. */
    $t^{\star,k+1} \leftarrow \tilde{t}^{\star,k}$;
    $t^{\star,\infty} \leftarrow t^{\star,k+1}$;
else
    $t^{\star,k+1} \leftarrow \alpha\tilde{t}^{\star,k} + (1-\alpha)t^{\star,k}$;
    /* $\alpha \in]0, 1[$ being a relaxation parameter given by the ICS. */
end if

bounded by the tolerance ϵ_{rel} of the scheme, i.e.

$$\frac{||\mathcal{M}_j^{\Delta t} \circ \mathcal{M}_i^{\Delta t}(\mathbf{b}_{ji}^{n+1,\infty}) - \mathbf{b}_{ji}^{n+1,\infty}||}{||\mathbf{b}_{ji}^{n+1,\infty}||} \leq \epsilon_{\text{rel}}.$$

- Model i and model j are synchronized on potential internal triggered events. In particular, they are synchronized on the time of the first event with an error also bounded by the tolerance ϵ_{rel}, i.e.

$$\frac{\left|t^{n+1} - t^{\star}\right|}{\Delta t} = \frac{|t^{\star,\infty} - t^{\star}|}{\Delta t} \leq \epsilon_{\text{rel}}$$

5. NUMERICAL ANALYSIS AND EXAMPLES

In the following, the aim is to present numerical solutions of some coupled LP models implemented in the CEA's PROCOR platform with both ECSs and ICSs. The industrial PROCOR platform Le Tellier et al. (2015) allows generic "black-box" physical models coupling solved by various ECSs and ICSs. It is dedicated to the fast robust setup of coupled problems taking the form of complex systems Boccara (2010) for the simulation of severe accidents in nuclear reactors. For computational efficiency reasons in this industrial context, one important constraint is that we want to be able to keep a sufficiently large coupling time step in comparison to the characteristic times of the physical phenomena being involved. With this constraint, it will be shown that even "simple" coupled problems of LP models may produce unexpected artifacts in terms of coupling and synchronization and how the use of ICSs to solve them can provide benefits.

For instance, let us consider the heat conduction between domains Ω_1 and Ω_2 calculated respectively by solvers $\mathcal{M}_1$ and $\mathcal{M}_2$. As depicted in Fig. 5, solvers are coupled at interface $\Gamma_{12} = \partial\Omega_1 \cap \partial\Omega_2$ of unit area $S = 1$ with Dirichlet-Neumann boundary conditions given by Eqs. 4, 5, and 6 or Eqs. 7, 8 and 9. This coupling ensure the "well-posedness" of the domain decomposition problem Dolean et al. (2015). For this coupled problem, two types of heat conduction solvers are used : LP solvers and finer solvers based on a space discretization of the 1D heat equation. Finally, we assume a cylindrical geometry in such a way that the length and mass of each domain are related with $m_{\{1,2\}} = \rho_{\{1,2\}} S e_{\{1,2\}} = \rho_{\{1,2\}} e_{\{1,2\}}$.

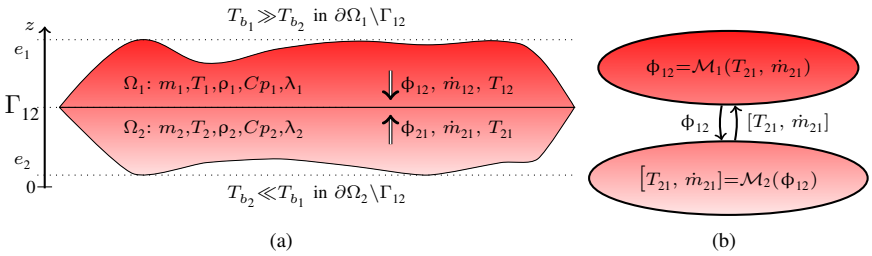


FIG. 5: Heat conduction between domains Ω_1 and Ω_2 : notations (*left*) and Dirichlet-Neumann coupling of solvers $\mathcal{M}_1$ and $\mathcal{M}_2$ (*right*).

5.1 Linear stability analysis of lumped parameter solver coupling

First, we consider the Dirichlet-Neumann boundary conditions at the fixed interface Γ_{12} given by Eqs. 7, 8 and 9 $\dot{m}_{12} = -\dot{m}_{21} = 0, T_{12} = T_{21}, \phi_{12} = -\phi_{21}$. The two domain masses are constant and only their energy conservation equations are calculated. In the following, we analyze the linear stability of the coupling of LP solvers. They are based on a time discretization of the energy conservation Eq. 2 and closure laws Eqs. 10 and 11 to solve the heat conduction problem. A similar stability analysis for finer 1D solvers can be found in Giles (1997).

5.1.1 Stability of a toy explicit coupling scheme

The aim here is to build a prototype of a 'toy' ECS with no subcycling and only one level of time discretization. Such coupling scheme should allow us to solve the coupled problem while solvers of domains Ω_1 and Ω_2 are only called once during a time step. To do so, we use an implicit Euler scheme for the first domain Ω_1 and an explicit Euler scheme for the second domain Ω_2 . Denoting by Δt the time step, the discretized equations for the first domain Ω_1 are given by

$$\begin{cases} \phi_{12}^n = \lambda_1 \dfrac{6T_1^n - 2T_{b_1} - 4T_{12}^n}{e_1} & \text{in } \Omega_1 \\ \rho_1 C_{p_1} e_1 \dfrac{T_1^n - T_1^{n-1}}{\Delta t} = -\phi_{12}^n & \text{in } \Omega_1 \end{cases} \tag{28}$$

with continuity of temperature at the interface, i.e.

$$T_{12}^n = T_{21}^n \quad \text{on } \Gamma_{12} \tag{29}$$

and for the second domain Ω_2

$$\begin{cases} T_{21}^{n+1} = -\dfrac{1}{4}\dfrac{e_2}{\lambda_2}\phi_{21}^n + \dfrac{3}{2}T_2^{n+1} - \dfrac{1}{2}T_{b_2} & \text{in } \Omega_2 \\ \rho_2 C_{p_2} e_2 \dfrac{T_2^{n+1} - T_2^n}{\Delta t} = -\phi_{21}^n & \text{in } \Omega_2 \end{cases} \tag{30}$$

with continuity of heat flux at the interface, i.e.

$$\phi_{21}^n = -\phi_{12}^n \quad \text{on } \Gamma_{12}. \tag{31}$$

The ECS is sequenced as follows : given an interface temperature T_{21}^n at time t^n, the first domain is advanced from time t^{n-1} to t^n and a heat flux ϕ_{12}^n is computed at time t^n which is then imposed to the second domain. It is then advanced from time t^n to t^{n+1} and it computes a new temperature T_{21}^{n+1} at time t^{n+1}, and so on. Thus, the coupling scheme only requires one call to each solver per time step so that the scheme can be called explicit.

We now analyze the linear stability of the coupling scheme. Combining Eqs. 28, 29, 30 and 31, we get

$$\left(1+6\frac{\Delta t}{\tau_1}\right)\phi_{12}^{n+1}-\left(1+\left(1-6\frac{\Delta t}{\tau_2}\right)\hbar\right)\phi_{12}^{n}+\hbar\,\phi_{12}^{n-1}=0 \tag{32}$$

with $\hbar=\frac{\lambda_1/e_1}{\lambda_2/e_2}$, the characteristic times of conduction $\tau_1=\rho_1 C_{p_1}{e_1}^2/\lambda_1$ in domain Ω_1 and $\tau_2=\rho_2 C_{p_2}{e_2}^2/\lambda_2$ in domain Ω_2. To ensure stability of the explicit scheme in domain Ω_2, Δt has to be small in comparison to τ_2, i.e. $\Delta t/\tau_2 \ll 1$. The simplified characteristic polynomial χ associated to the simplified second order linear difference Eq. 32 has two complex roots $x^\star_{\{1,2\}}$ which have to be of modulus strictly smaller than one in order to guaranty a stable coupling scheme.

We get the following results :

- When $\Delta t/\tau_1 \ll 1$ and $\hbar<1$, the two roots are real and a Taylor expansion gives

$$x_1^\star = 1-\frac{6}{1-\hbar}\frac{\Delta t}{\tau_1}+\mathcal{O}(\left(\frac{\Delta t}{\tau_1}\right)^2)$$
$$x_2^\star = \hbar+\frac{6\hbar^2}{1-\hbar}\frac{\Delta t}{\tau_1}+\mathcal{O}(\left(\frac{\Delta t}{\tau_1}\right)^2)$$

  When $\Delta t/\tau_1 \ll 1$ and $\hbar>1$, the two roots are complex of modulus

$$\left|x^\star_{\{1,2\}}\right|^2=\frac{\hbar}{1+6\frac{\Delta t}{\tau_1}}$$

- For any values of $\Delta t/\tau_1$, when $\hbar \ll 1$, roots of the characteristic polynomial χ are $x_1^\star \approx \frac{1}{1+6\frac{\Delta t}{\tau_1}}$ and $x_2^\star \approx \frac{\hbar}{1+6\frac{\Delta t}{\tau_1}}$ and when $\hbar \gg 1$, $x_1^\star \approx 1+\frac{6\frac{\Delta t}{\tau_1}}{\hbar}$ and $x_2^\star \approx \frac{\hbar}{1+6\frac{\Delta t}{\tau_1}}$.

Thus, in both cases when $\hbar>1$ the spurious solution for ϕ_{12} of Eq. 32 associated to root $x_2^\star$ increases with a growth rate of $\frac{\hbar}{1+6\,\Delta t/\tau_1}$. In most cases, the spurious solution will grow in time leading to an unstable scheme. On the contrary, cases when $\hbar<1$ give a stable coupling scheme.

Thus, the remaining question is the value $\hbar^{\text{crit}}$ of $\hbar$ above which the instabilities appear, thus defining the stability limit region of the ECS. Further calculations show that $\hbar^{\text{crit}}=\left|\frac{1+\,6\,\Delta t/\tau_1}{1-\,6\,\Delta t/\tau_2}\right|$ allowing to define a *pseudo* CFL condition for the linear stability of the ECS :

$$r_{12}(\Delta t)\overset{\text{def.}}{=}\left|\frac{1-\,6\,\Delta t/\tau_2}{1+\,6\,\Delta t/\tau_1}\right|\hbar<1 \tag{33}$$

Finally, the Dirichlet-Neumann explicit coupling should be done in such a way that the domain with the Dirichlet boundary condition, i.e. imposed boundary temperature, has the lower thermal conductivity or the higher characteristic length and the domain with the Neumann boundary condition, i.e. imposed boundary heat flux, has the higher thermal conductivity and the lower characteristic length. Thus, the stability limit region $\hbar^{\text{crit}}$ can be extended for larger values of $\Delta t/\tau_1$ which can be explained by the diffusive properties of the ICS used in domain Ω_1.

5.1.1.1 Numerical evidence of stability

This experimental analysis is illustrated in Fig. 6. At $t=0^-$, both domains are at an equilibrium temperature of $T_1=T_2=T_{21}=T_{b_1}=T_{b_2}=2000$ K. Boundary discontinuities are imposed at $t=0^+$, $T_{b_1}=3000$ K and $T_{b_2}=400$ K and at $t=3\,\tau_1$, $T_{b_1}=T_{b_2}=2000$ K. In addition, all of the computations use the value $\Delta t/\tau_2=1/100$ and $\Delta t/\tau_1=1/10$ corresponding to a stability limit region $\hbar^{\text{crit}}=1.6$. Fig. 6(a) shows that when $\hbar=\hbar^{\text{crit}}$ the oscillations created by the transient initiated by the boundary discontinuities are not damped but are stable while in

contrary in Fig. 6(c) with $\hbar < \hbar^{\text{crit}}$ the oscillations are well damped and in Fig. 6(d) with $\hbar > \hbar^{\text{crit}}$ the oscillations are clearly unstable. However, those three figures show that the discontinuities at boundaries are instantly propagated to the interface temperature and to the coupled domain in only one coupling time step. For comparison purpose, solution given by the coupling of the 1D heat equation solvers is given in Fig. 6(b). For the computation, the same physical values as for the coupling of LP solvers are used. The values meet the requirements expressed in Giles (1997) to ensure proper stability of such solvers. For this coupling, the discontinuities at boundaries start a much slower and smaller transient ($T_{21}(t = 1000\,\text{s}) \approx 2150$ K) than for the LP coupling ($T_{21}(t = 100\,\text{s}) \approx 2700$ K). The fact that LP models propagate in one coupling time step all their discontinuities to the other models tend to create fast and important transient. Thus, the use of ECS might not be adapted for this case.

5.1.2 Stability of a toy implicit coupling scheme

The iterative algorithm is described below. Given an initial boundary temperature $T_{21}^{n+1,0}$ for domain Ω_1, it iterates over $k \geq 0$ with the following steps :

1. Use an **implicit** solver in Ω_1 to find the interface heat flux $\phi_{12}^{n+1,k+1}$

$$\begin{cases} \phi_{12}^{n+1,k+1} = \lambda_1 \dfrac{6T_1^{n+1} - 2T_{b_1} - 4T_{12}^{n+1}}{e_1} & \text{in } \Omega_1 \\ \rho_1 C_{p_1} e_1 \dfrac{T_1^{n+1} - T_1^n}{\Delta t} = -\phi_{12}^{n+1,k+1} & \text{in } \Omega_1 \end{cases} \tag{34}$$

   with continuity of temperature at the interface, i.e.

$$T_{12}^{n+1} = T_{21}^{n+1,k} \quad \text{on } \Gamma_{12}. \tag{35}$$

2. Use an **implicit** solver in Ω_2 to find the interface temperature $\tilde{T}_{21}^{n+1,k+1}$

$$\begin{cases} \tilde{T}_{21}^{n+1,k+1} = -\dfrac{1}{4}\dfrac{e_2}{\lambda_2}\phi_{21}^{n+1} + \dfrac{3}{2}T_2^{n+1} - \dfrac{1}{2}T_{b_2} & \text{in } \Omega_2 \\ \rho_2 C_{p_2} e_2 \dfrac{T_2^{n+1} - T_2^n}{\Delta t} = -\phi_{21}^{n+1} & \text{in } \Omega_2 \end{cases} \tag{36}$$

   with continuity of heat flux at the interface, i.e.

$$\phi_{21}^{n+1} = -\phi_{12}^{n+1,k+1} \quad \text{on } \Gamma_{12}. \tag{37}$$

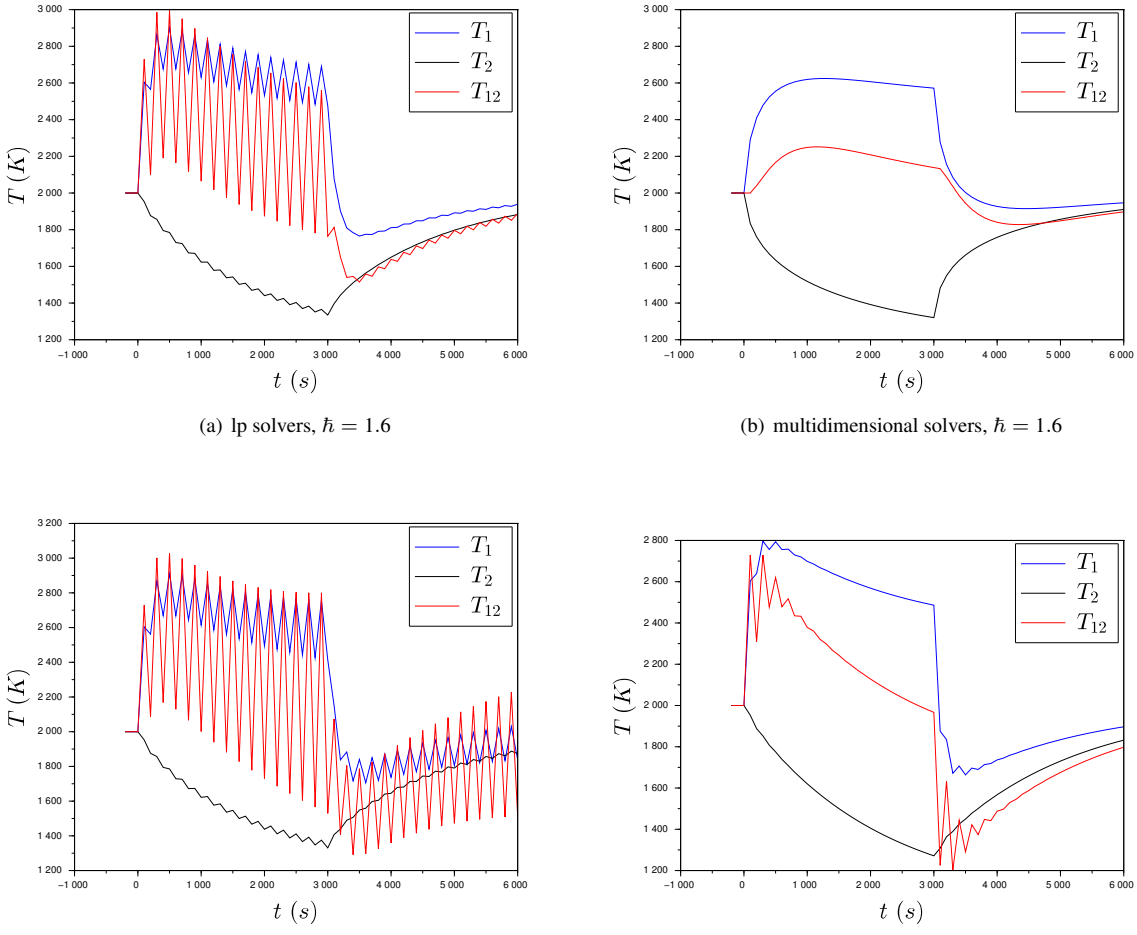
3. Test convergence level with

$$\frac{\left|\tilde{T}_{21}^{n+1,k+1} - T_{21}^{n+1,k}\right|}{T_{21}^{n+1,k}} \leq \epsilon_{\text{rel}}. \tag{38}$$

   If the previous predicate is not satisfied, relax the interface temperature by

$$T_{21}^{n+1,k+1} = \omega \tilde{T}_{21}^{n+1,k+1} + (1 - \omega) T_{21}^{n+1,k} \tag{39}$$

   and loop again. Otherwise, the final interface heat flux and temperature are given by $T_{21}^{n+1} = T_{21}^{n+1,k}$ and $\phi_{12}^{n+1} = \phi_{12}^{n+1,k}$. Consequently, internal variables T_1^{n+1} and T_2^{n+1} can be fully implicitly calculated : the iterative algorithm effectively allows to use implicit solvers for the two domains while allowing to decouple the two domains.



(a) lp solvers, $\hbar = 1.6$

(b) multidimensional solvers, $\hbar = 1.6$

(c) $\hbar = 1.63$

(d) $\hbar = 1.0$

FIG. 6: Propagation of discontinuities at boundaries in the explicit coupling of lumper parameter and finer (upper right) solvers for heat conduction for different values of $\hbar$ with $\Delta t/\tau_2 = 1/100$ and $\Delta t/\tau_1 = 1/10$.

Combining Eqs. 34, 35, 36 and 37, simple calculations lead to the following equation giving the interface temperature iterates $T_{21}^{n+1,k}$

$$T_{21}^{n+1,k+1} = \left[1 - \left(1 + \frac{1 - 6\frac{\Delta t}{\tau_2}}{1 + 6\frac{\Delta t}{\tau_1}}\hbar\right)\omega\right] T_{21}^{n+1,k} + g(T_1^n, T_2^n, T_{b_1}, T_{b_2}). \tag{40}$$

We note T_{21}^{n+1} the solution of the fixed point equation associated with the previous equation and $e_k = \left|T_{21}^{n+1,k} - T_{21}^{n+1}\right|$ the error at iteration k. Then iteration errors are given by equation

$$e_{k+1} = \left|1 - \left(1 + \frac{1 - 6\frac{\Delta t}{\tau_2}}{1 + 6\frac{\Delta t}{\tau_1}}\hbar\right)\omega\right| e_k. \tag{41}$$

Thus, the iterative algorithm converges toward the interface solutions, i.e. $T_{21}^{n+1,\infty} = T_{21}^{n+1}$ and $\phi_{12}^{n+1,\infty} = \phi_{12}^{n+1}$, if and only if

$$0 < \omega < \frac{2}{1 + \frac{\left|1-6\frac{\Delta t}{\tau_2}\right|}{\left|1+6\frac{\Delta t}{\tau_1}\right|}\hbar} = \frac{2}{1 + r_{12}(\Delta t)}. \tag{42}$$

Note that when $\frac{\Delta t}{\tau_1} \ll 1$ and $\frac{\Delta t}{\tau_2} \ll 1$, condition Eq. 42 reads

$$0 < \omega < \frac{2}{1 + \hbar}. \tag{43}$$

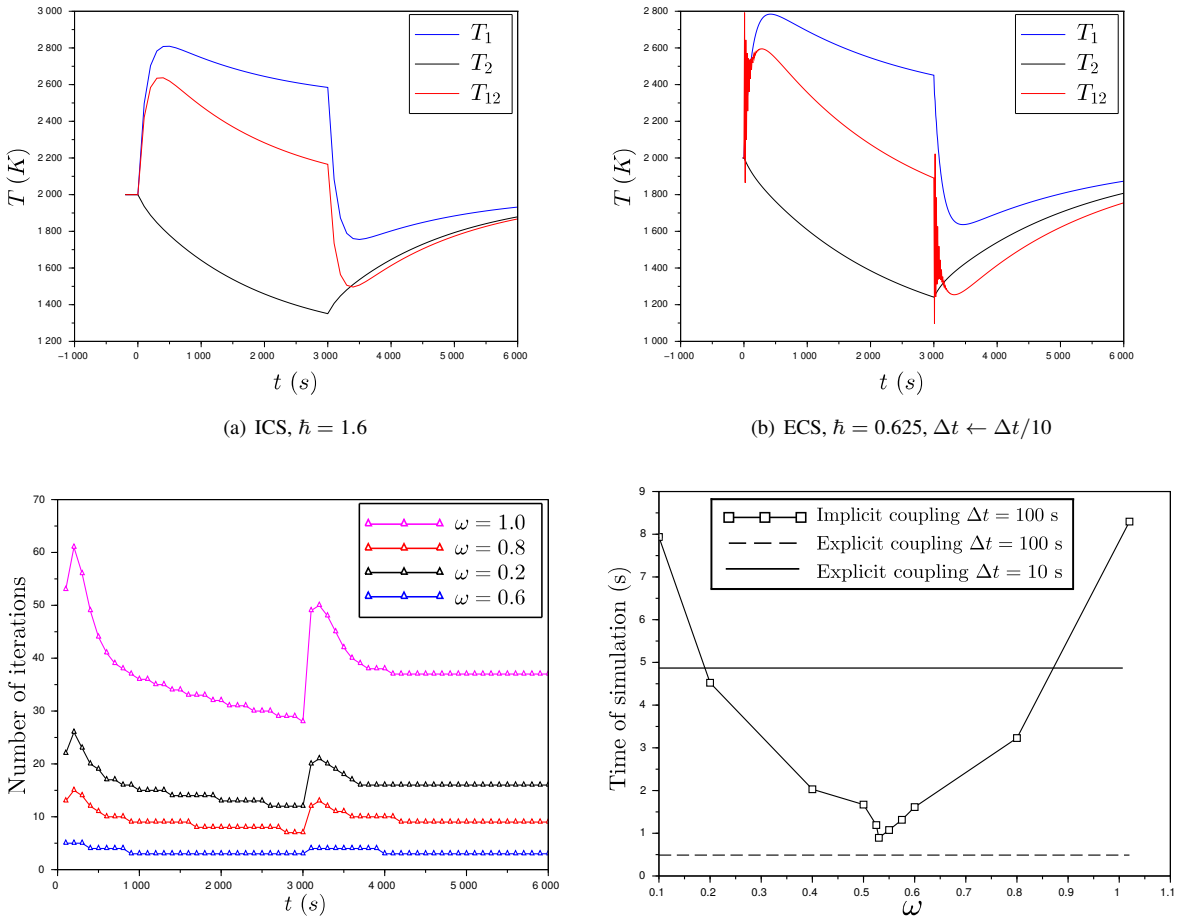
In cases where $\hbar > 1$ for which the ECS diverges, the iterative algorithm needs $\omega < 1$, i.e. under-relaxation to converge, which might lead to slow convergence.

5.1.2.1 Numerical evidence of stability

The experimental stability analysis is illustrated in Fig. 7. As in Sec. 5.1.1, both domains are at an equilibrium temperature of 2000 K and the same discontinuities at boundaries are occurring at the same time $t = 0^+$ and $t = 3\,\tau_1$. For the computations, we still use $\Delta t/\tau_2 = 1/100$, $\Delta t/\tau_1 = 1/10$. We use $\hbar = 1.6$ corresponding to the stability limit region for the ECS ($\hbar = \hbar^{\text{crit}} = 1.6$) for which this scheme is showing constant oscillations. For this value, the corresponding maximal value of ω to ensure convergence of the iterative process is $\omega \approx 1.03$. The relative tolerance used is $\epsilon_{\text{rel}} = 10^{-4}$. Several computations confirm the predicted behavior of the ICS : as long as ω stays under the calculated maximum value 1.03, the ICS converges toward the same smooth solution given by Fig. 7(a). Even for values $\hbar > \hbar^{\text{crit}}$, the ICS handles smoothly the fast dynamics and important transient initiated by the instant propagation of the discontinuities at boundaries in the two domains by the LP models. For cases where the ECS is not unconditionally unstable, i.e. $r_{12}(\Delta t) < 1$, one way to achieve stability is to reduce the coupling time step : Fig. 7(b) shows an explicit coupling solution with $\hbar = 0.625 < \hbar^{\text{crit}} \approx 1$ and coupling time step Δt reduced by a factor of 10. However, while effectively reducing the oscillations, using an ECS with such a small coupling time step leads to higher computational time. The ICS allows to keep a sufficiently large coupling time step and a limited number of iterations as shown in Figs. 7(c) and 7(d). In the last figure, it is important to understand that only the solution given by the ECS with a small coupling time step is usable but is costly to obtain while the solution given with the ICS is good and costs 5 times less. For this problem, the optimal relaxation parameter leading to the higher convergence rate can be analytically calculated from Eq. 41 and is given by $\omega = 1/(1 + r_{12}(\Delta t)) \approx 0.52$. However, most of the time the optimal relaxation parameter is highly dependent to the problem.

5.2 Time scheduling: synchronization of internal events

For instance, let us describe the state of domain Ω_2 into a graph of three states as depicted in Fig. 8 : *Heating*, *Melting* and *Empty* states. A state transition occurs when a certain algebraic function is activated, e.g. $\dot{m}_{21} < 0$. As explained



(a) ICS, $\hbar = 1.6$

(b) ECS, $\hbar = 0.625$, $\Delta t \leftarrow \Delta t/10$

(c) ICS number of iterations, $\hbar = 1.6$

(d) Schemes time of simulation, $\hbar = 1.6$

FIG. 7: Propagation of discontinuities at boundaries of lumper parameter heat conduction coupled solvers solved with an ICS (upper left) and an ECS (upper right) with much smaller coupling time step.

FIG. 8: Internal states of domain Ω_2 and their transition functions.

in Sec. 4.2, each state has its own solver, its own set of equations and its own interface and boundary conditions. State transitions can lead to discontinuities of state or interface variables. For this case, equations of solver $\mathcal{M}^{\text{Heating}}$ are the ones described previously in Sec. 5.1 while solver $\mathcal{M}^{\text{Melting}}$ continuous equations are given by

$$\begin{cases} \phi_{21} = \lambda_1 \dfrac{6T_2 - 4T_{21} - 2T_{b_2}}{e_2} & \text{in } \Omega_2 \\ \dfrac{\mathrm{d}m_2}{\mathrm{d}t} = \rho_2 \dfrac{\mathrm{d}e_2}{\mathrm{d}t} = -\dot{m}_{21} & \text{in } \Omega_2 \\ \rho_2 C_{p_2} e_2 \dfrac{\mathrm{d}T_2}{\mathrm{d}t} = -\phi_{21} & \text{in } \Omega_2, \end{cases} \tag{44}$$

leading to a mobile fusion solidification front at interface Γ_{12} given by Eqs. 4, 5 and 6 $\phi_{12} = -\phi_{21} + \Delta\mathcal{H}^{\text{fus.}}\dot{m}_{21}$, $\dot{m}_{12} = -\dot{m}_{21}$ and $T_{12} = T_{21} = T^{\text{fus.}}$. Again, another set of internal and boundary equations are used for solver $\mathcal{M}^{\text{Empty}}$ leading to a fixed interface given by Eqs. 7, 8 and 9 $\dot{m}_{12} = -\dot{m}_{21} = 0, T_{12} = T_{21}, \phi_{12} = -\phi_{21}$.

Again, at $t = 0^-$, both domains are at an equilibrium temperature of $T_1 = T_2 = T_{21} = T_{b_1} = T_{b_2} = 2000$ K. In particular, domain Ω_2 is in *Heating* state. At $t = 0^+$, $T_{b_1} = 3000$ K and $T_{b_2} = 3000$ K to force the *Melting* state transition when T_{21} reaches $T^\star = 2100$ K at time $t^\star_{H\to M}$. Once this state has been reached, the second domain is in *Melting* state until only a residual mass $m_2 = m^\epsilon = 150$ kg is left (corresponding to height $e_2 = 1.5$ cm) and the *Empty* state is reached at time $t^\star_{M\to E}$. The computation then stops when domain Ω_1 reaches a near stationary state. In addition, all of the computations use the value $\tau_2 = 10^4$ s, $\tau_1 = 8 \times 10^3$ s and different values for the time step Δt. The chosen values ensure that the ECS is stable, i.e. $r_{12}(\Delta t) < 1.0$. Table 1 shows the times of the internal events of domain Ω_2 computed by the ECS and the ICS for several coupling time steps. Usually, we ask the coupling time step to be bounded by

$$\tau/10 = 800 \text{ s} \geq \Delta t \geq \tau/100 = 80 \text{ s} \quad \text{with} \quad \tau = \min(\tau_1, \tau_2)$$

for computational efficiency : in the table, only $\Delta t = 100$ s satisfies this criterion. In comparison to the reference solution (marked with †) given by the ECS for a small time step $\Delta t = 1$ s for which the scheme has converged, the ICS is able to predict the state transition with high accuracy and adapt its time step to synchronize both domains when the event occurs. However, the synchronization mechanism of the ECS only allows the domains to be informed of an event at the end of the time step. During this window of time, both domains are in a non physical state, i.e. the domain Ω_1 is not aware of the disappearance of the domain Ω_2. This leads to numerical errors of order $\mathcal{O}(\Delta t)$ and the numerical creation of either mass and/or energy leading to different stationary states for the ECS. For a time step $\Delta t = 100$ s respecting the constraint, the explicit solution leads to a relative error of 18% in term of mass. Reducing the coupling time step to 10 s leads to a relative error of 0.3% but, as seen previously, increases computational time. Besides, the ICS always converges toward the reference solution and the right stationary state even for large coupling time step.

This very simple example of coupling of LP models highlights the main drawback of such lightweight modeling : deleting spatial dependency in the equations force each model to instantly propagate all its data, e.g. its discontinuities or its state transitions, thus creating high and important transient or bringing the system in non physical states. Fixes are used to bring back the system in a coherent state which should be avoided at all cost. These phenomenons create numerical errors in the coupling which can be measured in term of numerical energy created at the interface Γ_{12}. We follow the same methodology as in Piperno and Farhat (2001). The local variation of energy $\Delta E^{n\to n+1}$ at interface Γ_{12} between times t^n and t^{n+1} is the sum of the energy $\Delta E_1^{n\to n+1}$ send by domain Ω_1 through the interface and the energy $\Delta E_2^{n\to n+1}$ send by domain Ω_2 through the interface. For the continuous case in which both domains are

	Δt (s)	**100**	50	25	10	1
	$r_{12}(\Delta t)$	< 1.0				
ECS	m_1^∞ (kg)	**1492**	1413	1347	1260	1256 †
	T_1^∞ (K)	**1710**	1708	1705	1702	1700 †
ICS	m_1^∞ (kg)	**1255**	1255			
	T_1^∞ (K)	**1700**	1700			
ECS	$t^\star_{H\to M}$ (s)	**1700**	1600	1600	1590	1584 †
	$t^\star_{M\to E}$ (s)	**3200**	3200	3150	3120	3117 †
ICS	$t^\star_{H\to M}$ (s)	**1583**	1583			
	$t^\star_{M\to E}$ (s)	**3120**	3117	3117	3116	3116

TABLE 1: Times $t^\star_{H\to M}$ and $t^\star_{M\to E}$ of domain Ω_2 internal events and stationary state m_1^∞ and T_1^∞ reached by domain Ω_1. Reference solution marked with †, solutions respecting the coupling time step constraint ($\Delta t \geq 80$ s) in blue and **bold**, otherwise in red.

strongly coupled, the local variation of energy is null. It is defined by equation

$$\begin{aligned}\Delta E^{n\to n+1} &= \Delta E_1^{n\to n+1} \; - \; (-\Delta E_2^{n\to n+1}) \\ &= \int_{t^n}^{t^{n+1}} \phi_{12} - \left(\int_{t^n}^{t^{n+1}} \phi_{21} - \Delta\mathcal{H}^{\text{fus.}} \dot{m}_{21} \right) \\ &= 0.\end{aligned}$$

The global energy $\Delta E^{0\to n}$ through interface Γ_{12} is defined by

$$\Delta E^{0\to n} = \sum_{k=0}^{n} \Delta E^{k\to k+1} = 0 \tag{45}$$

and is also null in the continuous case. For each energy variation ΔE, we define the relative energy

$$\epsilon(\Delta E) = \frac{\Delta E}{E^\star} = \frac{\Delta E}{m_2(t^0) C_{p_2}(T^{\text{fus.}} - T_2(t^0)) + \Delta\mathcal{H}^{\text{fus.}} m_2(t^0)}$$

by the ratio of the energy ΔE to the energy $E^\star$ needed to heat the initial domain Ω_2 to its fusion temperature ($m_2(t^0)C_{p_2}(T^{\text{fus.}} - T_2(t^0))$) and to melt it ($\Delta\mathcal{H}^{\text{fus.}} m_2(t^0)$). With the physical values used, we have $E^\star \approx 10^8$ J.

When discretized for the ECS, the local variation of energy becomes

$$\Delta E^{n\to n+1} \approx \phi_{12}(t^{n+1})\Delta t \; - \; \left(\phi_{21}(t^n) - \Delta\mathcal{H}^{\text{fus.}} \dot{m}_{21}(t^n)\right) \Delta t$$

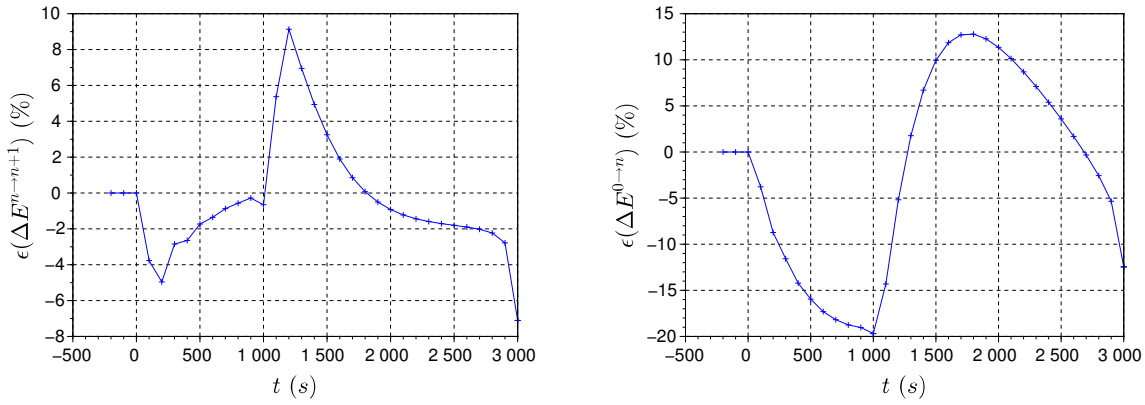
which emphasizes the "time-lag" between the two domains. Fig. 9(a) shows that the ECS can create locally 2.5% of $E^\star$, i.e. 2.5 10^6 J. Fig. 9(b) shows first that the global energy is not null and that terms of the sum Eq. 45 do not sum to zero. Besides, the explicit scheme can create globally more than 6% of $E^\star$, i.e. 6 10^6 J which can have disastrous effects on the whole accuracy.

For the ICS, the local variation of energy becomes

$$\Delta E^{n\to n+1} \approx \phi_{12}(t^{n+1,\infty})\Delta t \; - \; \left(\phi_{21}(t^{n+1,\infty}) - \Delta\mathcal{H}^{\text{fus.}} \dot{m}_{21}(t^{n+1,\infty})\right) \Delta t$$

in which variables with superscript $^\infty$ represent the value reached when the convergence criterion of the iterative scheme is satisfied. At convergence, we can bound the residual at interface in such a way that

$$\left|\phi_{12}(t^{n+1,\infty}) \; - \; \left(\phi_{21}(t^{n+1,\infty}) - \Delta\mathcal{H}^{\text{fus.}} \dot{m}_{21}(t^{n+1,\infty})\right)\right| \Delta t \leq \epsilon_{\text{rel}} \left|\phi_{12}(t^{n+1,\infty})\right| \Delta t.$$



(a) (b)

FIG. 9: Relative local energy variation (*left*) and global energy variation (*right*) at interface for the ECS

This allows us to bound the relative local energy at interface of the ICS with

$$\left|\epsilon(\Delta E^{n\to n+1})\right| = \left|\frac{\Delta E^{n\to n+1}}{E^\star}\right| \leq \epsilon_{\text{rel}}\ \frac{\left|\phi_{12}(t^{n+1,\infty})\right|\Delta t}{E^\star}$$

and the relative global energy with

$$\left|\epsilon(\Delta E^{0\to n})\right| \leq \sum_{k=0}^{n}\left|\frac{\Delta E^{k\to k+1}}{E^\star}\right| \leq n\ \epsilon_{\text{rel}}\ \frac{\max\limits_k\left|\phi_{12}(t^{k+1,\infty})\right|\Delta t}{E^\star}.$$

This shows that the imbalance of energy at interface of the ICS can be controlled with the relative tolerance ϵ_{rel}. This is confirmed by numerical experiments presented in Fig. 10(a) and in Fig.10(b) showing respectively the relative local and global energy at interface with a relative tolerance set to 10^{-4}. The maximum heat flux $\max_k\left|\phi_{12}(t^{k+1,\infty})\right|$ reached at the interface with unit area is equal to $\approx 1.5\times 10^5$ W.m^{-2} leading to a bound for the relative local energy at interface of 1.5×10^{-3} (expressed in percent in Fig. 10(a)).

6. CONCLUDING REMARKS AND PERSPECTIVES

In this paper, we have presented problems of coupled LP models with time management of state change events. In comparison to finer modeling approaches (e.g. mesh based modelings), this approach allows for fast calculations which are needed when doing statistical studies with a large number of calculations (e.g. design of computer experiments, Monte-Carlo methods). This is typically the true industrial context of modeling and simulation of severe accidents in nuclear reactors. However, we have shown that the LP modeling approach often leads to coupled problems with stiff, important and fast transients which are not suitable for being solved by ECSs. Thus, ICSs were presented and designed for proper events detection and models synchronization allowing to obtain stable and accurate solutions of coupled problems of lumped parameter models.

We proposed to study theoretically the numerical stability of ECSs and ICSs on coupled problems of LP models representing the heat conduction and ablation between a liquid subdomain and a solid subdomain. The interface equation between the coupled subdomains consists of a non-linear Stefan problem involving the solid temperature, the liquid temperature and the phase boundary location. Such coupled problems are of prime interest in SAs when

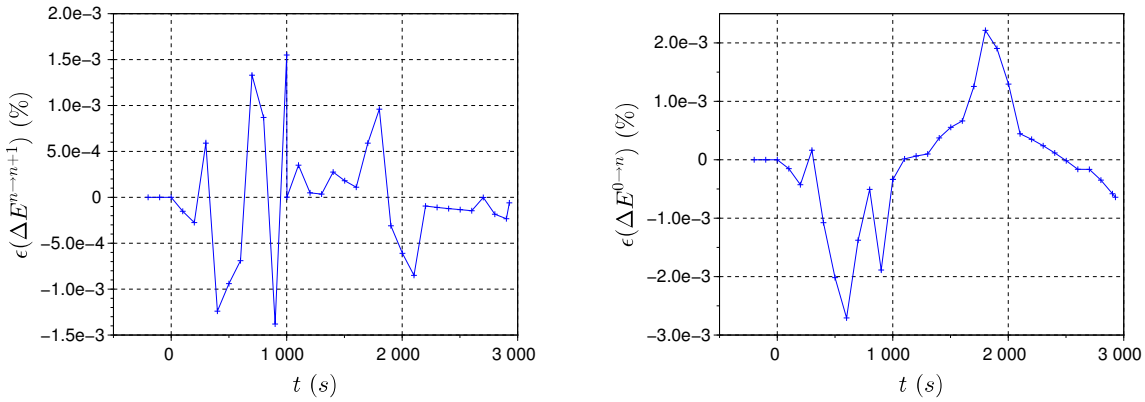


FIG. 10: Relative local energy variation (*left*) and global energy variation (*right*) at interface for the ICS

considering the melting of cladding materials, the melting of the reactor pressurized vessel (in particular in the context of the In-Vessel-Retention strategy Carénini et al. (2018); Theofanous et al. (1997)) or the melting of the concrete reactor pit (Molten Core Concrete Interaction Spindler et al. (2006)). From this short study we have learned that Dirichlet-Neumann boundary conditions should be set up carefully in order to ensure stability of the coupling schemes. However with industrial constraints on the coupling time step to ensure fast calculations, ECSs remain very unstable and cannot be reasonable candidates in general cases to give good and precise results. Besides, even the cheapest ICSs that only use relaxation can give precise and stable calculations leading to trustworthy results. They turn out to be very interesting in term of computational times in comparison to ECSs with a highly reduced coupling time step to ensure stability. Furthermore, the designed ICSs are able to predict events and discontinuities in the coupled models allowing synchronization between them. This tends to suggest that industrial problems of coupled lumped parameter models could substantially benefit from the use of implicit coupling schemes.

In this work, only the effects of heat conduction and ablation in pure body (the liquid and solid subdomains) are considered. Additional physical assumptions are made allowing us to carry out numerical analyses. The obtained results give us a good insight of the proper way to couple liquid and solid solvers with a Stefan interface. Yet, these assumptions are still representative of SAs in LWRs. Indeed, the considered coupled problem of pure body is relevant when considering the ablation of the reactor pressurized vessel since the liquidus and solidus temperatures of steel remain very close in such a way that, on each side of the fusion front propagating in the vessel, the pure body assumption is valid. However, this is not the case for a fusion front in the corium pool. In an on-going work in the frame of the `PROCOR` platform, enthalpy-temperature relations based on CALPHAD calculations Tiwari and Le Tellier (2017) are used to take into account the coupling between thermochemistry phenomena, e.g. the composition of the liquid and solid subdomains, with thermohydraulic phenomena, e.g. heat transfer induced by natural convection. While still being under development, the resulting more complex coupled problem between the two subdomains can be solved very precisely with ICSs when the coupling is done with respect to conditions found in Sec.5. Furthermore, in addition to these composition effects considerations, we could add a model to take into account the metal-water oxidation and hydrogen generation during the melting of cladding materials Haurais (2016).

In future developments, it would be worthwhile to add some *artificial intelligence* to the coupling process (either by rules or by some machine learning process). For instance, depending on the strength of a coupling, i.e. the value of the residual at interfaces, the coupling scheme should be able to choose between an ECS or an ICS to avoid using potentially costly iterative scheme. Indeed, we have seen that such schemes can be optimized to ensure fast calculations but this is still highly problem-dependent. In particular this can be very useful in the context of statistical

studies where the experiments are run into an automated process. Finally, if ICS are required to ensure proper stability of the whole system but are still too computationally expensive, time parallelism techniques like *parareal* methods Maday and Turinici (2005) could be used.

ACKNOWLEDGMENTS

This work has been carried out within the framework of the PROCOR platform development funded by CEA, EDF and FRAMATOME.

REFERENCES

Aitken, A.C., XXV.—On Bernoulli's Numerical Solution of Algebraic Equations, *Proceedings of the Royal Society of Edinburgh*, vol. **46**, pp. 289–305, 1927.
URL https://www.cambridge.org/core/journals/proceedings-of-the-royal-society-of-edinburgh/article/xxvon-bernoullis-numerical-solution-of-algebraic-equations/64D4A7C56F1EFEC696AF68D7870DB451

Boccara, N., *Modeling Complex Systems*, Graduate Texts in Physics, Springer New York, New York, NY, 2010.

Bonnet, J.M. and Seiler, J.M., Thermohydraulic phenomena in corium pool: the BALI experiment, *Proc. of ICONE 7*, Tokyo, Japan, 1999.

Carénini, L., Fichot, F., and Seignour, N., Modelling issues related to molten pool behaviour in case of in-vessel retention strategy, *Annals of Nuclear Energy*, vol. **118**, pp. 363–374, 2018.
URL http://www.sciencedirect.com/science/article/pii/S0306454918302172"

Causin, P., Gerbeau, J.F., and Nobile, F., Added-mass effect in the design of partitioned algorithms for fluid–structure problems, *Computer Methods in Applied Mechanics and Engineering*, vol. **194**, no. 42–44, pp. 4506–4527, 2005.
URL http://www.sciencedirect.com/science/article/pii/S0045782504005328

Degroote, J., Bathe, K.J., and Vierendeels, J., Performance of a new partitioned procedure versus a monolithic procedure in fluid–structure interaction, *Computers & Structures*, vol. **87**, no. 11-12, pp. 793–801, 2009.
URL http://linkinghub.elsevier.com/retrieve/pii/S0045794908002605

Degroote, J. and Vierendeels, J., Multi-level quasi-Newton coupling algorithms for the partitioned simulation of fluid–structure interaction, *Computer Methods in Applied Mechanics and Engineering*, vol. **225–228**, pp. 14–27, 2012.
URL http://www.sciencedirect.com/science/article/pii/S0045782512000862

Dolean, V., Jolivet, P., and Nataf, F., *An Introduction to Domain Decomposition Methods*, Other Titles in Applied Mathematics, Society for Industrial and Applied Mathematics, 2015.

Farhat, C., Lesoinne, M., and Maman, N., Mixed explicit/implicit time integration of coupled aeroelastic problems: Three-field formulation, geometric conservation and distributed solution, *International Journal for Numerical Methods in Fluids*, vol. **21**, no. 10, pp. 807–835, 1995.

Farhat, C., Park, K.C., and Dubois-Pelerin, Y., An unconditionally stable staggered algorithm for transient finite element analysis of coupled thermoelastic problems, *Computer methods in applied mechanics and engineering*, vol. **85**, no. 3, pp. 349–365, 1991.
URL http://www.sciencedirect.com/science/article/pii/004578259190102C

Farhat, C., Rallu, A., Wang, K., and Belytschko, T., Robust and provably second-order explicit–explicit and implicit–explicit staggered time-integrators for highly non-linear compressible fluid–structure interaction problems, *International Journal for Numerical Methods in Engineering*, vol. **84**, no. 1, pp. 73–107, 2010.

Farhat, C. and Sobh, N., A consistency analysis of a class of concurrent transient implicit/explicit algorithms, *Computer methods in applied mechanics and engineering*, vol. **84**, no. 2, pp. 147–162, 1990.
URL http://www.sciencedirect.com/science/article/pii/0045782590901142

Felippa, C.A. and Park, K.C., Staggered transient analysis procedures for coupled mechanical systems: formulation, *Computer Methods in Applied Mechanics and Engineering*, vol. **24**, no. 1, pp. 61–111, 1980.
URL http://www.sciencedirect.com/science/article/pii/0045782580900407

Ganine, V., Hills, N.J., and Lapworth, B.L., Nonlinear acceleration of coupled fluid–structure transient thermal problems by Anderson mixing, *International Journal for Numerical Methods in Fluids*, vol. **71**, no. 8, pp. 939–959, 2013.

Gerbeau, J.F. and Vidrascu, M., A Quasi-Newton Algorithm Based on a Reduced Model for Fluid-Structure Interaction Problems in Blood Flows, *ESAIM: Mathematical Modelling and Numerical Analysis*, vol. **37**, no. 4, pp. 631–647, 2003.

Giles, M.B., Stability analysis of numerical interface conditions in fluid-structure thermal analysis, *International Journal for Numerical Methods in Fluids*, vol. **25**, no. 4, pp. 421–436, 1997.

Guillard, H. and Farhat, C., On the significance of the geometric conservation law for flow computations on moving meshes, *Computer Methods in Applied Mechanics and Engineering*, vol. **190**, no. 11–12, pp. 1467–1482, 2000.
URL http://www.sciencedirect.com/science/article/pii/S0045782500001730

Haurais, F., Evaluate the contribution of the fuel cladding oxidation process on the hydrogen production from the reflooding during a potential severe accident in a nuclear reactor, Theses, Université Paris-Saclay, 2016.

Jacquemain, D., *Les accidents de fusion du coeur des réacteurs nucléaires de puissance: état des connaissances (French)*, Collection sciences et techniques, EDP sciences, Les Ulis, 2013.

Joosten, M.M., Dettmer, W.G., and Perić, D., Analysis of the block Gauss-Seidel solution procedure for a strongly coupled model problem with reference to fluid-structure interaction, *International Journal for Numerical Methods in Engineering*, vol. **78**, no. 7, pp. 757–778, 2009.

Kassiotis, C., Ibrahimbegovic, A., Niekamp, R., and Matthies, H.G., Nonlinear fluid–structure interaction problem. Part I: implicit partitioned algorithm, nonlinear stability proof and validation examples, *Computational Mechanics*, vol. **47**, no. 3, pp. 305–323, 2011.

Küttler, U. and Wall, W.A., Fixed-point fluid–structure interaction solvers with dynamic relaxation, *Computational Mechanics*, vol. **43**, no. 1, pp. 61–72, 2008.

Le Tellier, R., Saas, L., and Payot, F., Phenomenological analyses of corium propagation in LWRs: the PROCOR software platform, *Proc. of the 7th European Review Meeting on Severe Accident Research ERMSAR-2015*, Marseille, France, 2015.

Le Tellier, R., Skrzypek, E., and Saas, L., On the treatment of plane fusion front in lumped parameter thermal models with convection, *Applied Thermal Engineering*, vol. **120**, pp. 314–326, 2017.
URL http://www.sciencedirect.com/science/article/pii/S1359431116327119

Maday, Y. and Turinici, G., 2005. The Parareal in Time Iterative Solver: a Further Direction to Parallel Implementation. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 441–448.

Mao, G. and Petzold, L.R., Efficient integration over discontinuities for differential-algebraic systems, *Computers & Mathematics with Applications*, vol. **43**, no. 1, pp. 65–79, 2002.
URL http://www.sciencedirect.com/science/article/pii/S0898122101002723

Mehl, M., Uekermann, B., Bijl, H., Blom, D., Gatzhammer, B., and van Zuijlen, A., Parallel coupling numerics for partitioned fluid–structure interaction simulations, *Computers & Mathematics with Applications*, vol. **71**, no. 4, pp. 869–891, 2016.
URL http://linkinghub.elsevier.com/retrieve/pii/S0898122115005933

Michler, C., van Brummelen, E.H., and de Borst, R., An interface Newton-Krylov solver for fluid-structure interaction, *International Journal for Numerical Methods in Fluids*, vol. **47**, no. 10-11, pp. 1189–1195, 2005.

Minami, S. and Yoshimura, S., Performance evaluation of nonlinear algorithms with line-search for partitioned coupling techniques for fluid-structure interactions, *International Journal for Numerical Methods in Fluids*, vol. **64**, no. 10-12, pp. 1129–1147, 2010.

Petzold, L., Differential/Algebraic Equations are not ODE's, *SIAM Journal on Scientific and Statistical Computing*, vol. **3**, no. 3, pp. 367–384, 1982.

Piperno, S. and Farhat, C., Partitioned procedures for the transient solution of coupled aeroelastic problems–Part II: energy transfer analysis and three-dimensional applications, *Computer methods in applied mechanics and engineering*, vol. **190**, no. 24, pp. 3147–3170, 2001.
URL http://www.sciencedirect.com/science/article/pii/S0045782500003868

Piperno, S., Farhat, C., and Larrouturou, B., Partitioned procedures for the transient solution of coupled aroelastic problems Part I: Model problem, theory and two-dimensional application, *Computer Methods in Applied Mechanics and Engineering*, vol. **124**, no. 1–2, pp. 79 – 112, 1995.
URL http://www.sciencedirect.com/science/article/pii/0045782595927079

Ramière, I. and Helfer, T., Iterative residual-based vector methods to accelerate fixed point iterations, *Computers & Mathematics*

with Applications, vol. **70**, no. 9, pp. 2210–2226, 2015.
URL http://linkinghub.elsevier.com/retrieve/pii/S0898122115004046

Rempe, J.L., Knudson, D.L., Cebull, M., and Atwood, C.L., Potential for in-vessel retention through ex- vessel flooding, *Proc. of the OECD/CSNI Workshop on In-Vessel Core Debris Retention and Coolability*, 1998.

Sehgal, B.R., *Nuclear safety in Light Water Reactors: Severe Accident Phenomenology*, 1st Edition, Elsevier/Academic Press, Amsterdam ; Boston, 2012.

Spindler, B., Tourniaire, B., and Seiler, J.M., Simulation of MCCI with the TOLBIAC-ICB code based on the phase segregation model, *Nuclear Engineering and Design*, vol. **236**, pp. 2264–2270, 2006.

Theofanous, T.G., Liu, C., Addition, S., Angelini, S., Kymalainen, O., and Salimassi, T., In-vessel coolability and retention of a core melt, *Nuclear Engineering and Design*, vol. **169**, pp. 1–48, 1997.

Tiwari, V. and Le Tellier, R., On the use of CALPHAD-based enthalpy-temperature relations in suboxidized corium plane front solidification modelling, *Proc. of the 8th European Review Meeting on Severe Accident Research ERMSAR-2017*, 2017.

Vierendeels, J., Lanoye, L., Degroote, J., and Verdonck, P., Implicit coupling of partitioned fluid–structure interaction problems with reduced order models, *Computers & Structures*, vol. **85**, no. 11–14, pp. 970–976, 2007.
URL http://www.sciencedirect.com/science/article/pii/S0045794906003865

Zhang, L., Zhou, Y., Zhang, Y., Tian, W., Qiu, S., and Su, G., Natural convection heat transfer in corium pools: A review work of experimental studies, *Progress in Nuclear Energy*, vol. **79**, pp. 167–181, 2015.
URL http://www.sciencedirect.com/science/article/pii/S014919701400328X

4.2 Problème couplé de la vidange d'une couche

4.2.1 Équation d'évolution et problème couplé

On considère un matériau m décrit par une masse, une température et une composition. Comme décrit 4.1, sont ajoutés ou retirés à ce matériau un ensemble de débits $\dot{\mathsf{m}}_i$ pour $1 \leq i \leq l$.

L'évolution du matériau m entre deux instants t^n et t^{n+1} est donnée par l'équation

$$\mathsf{m}(t^{n+1}) = \mathsf{m}(t^n) + \sum_{1 \leq i \leq l} \int_{t^n}^{t^{n+1}} \dot{\mathsf{m}}_i(\tau) \mathrm{d}\tau. \tag{4.1}$$

Ces débits peuvent avoir une toute autre température ou une toute autre composition. De ce fait, l'opération d'ajout, symbolisée par le symbole + dans (4.1), peut potentiellement être complexe.

Le débit $\dot{\mathsf{m}}_i$ pour $1 \leq i \leq l$ peut être indépendant du matériau et être imposé par l'extérieur. Cependant, il peut aussi être calculé à partir de l'état de celui-ci. C'est le cas par exemple lorsque le débit correspond à une partie du matériau rentrant en fusion sous l'effet d'un flux de chaleur. À chaque instant, $\dot{\mathsf{m}}_i$ dépend de m et donc, d'après l'équation (4.1), dépend aussi des autres débits $\dot{\mathsf{m}}_j$ pour $j \neq i$. Au débit $\dot{\mathsf{m}}_i$ est associé un solveur $\mathcal{M}_i$ donné par l'équation

$$\dot{\mathsf{m}}_i = \mathcal{M}_i(\dot{\mathsf{m}}_1, \ldots, \dot{\mathsf{m}}_{i-1}, \dot{\mathsf{m}}_{i+1}, \ldots, \dot{\mathsf{m}}_l) \tag{4.2}$$

dans laquelle les couplages entre débits apparaissent clairement. Le solveur $\mathcal{M}_i$ correspond à une physique particulière, par exemple un solveur de thermique du matériau calculant un débit sortant de matériau rentré en fusion comme évoqué précédemment.

L'équation d'évolution du matériau m (4.1) nécessite la connaissance de tous les débits $\dot{\mathsf{m}}_i$ pour $1 \leq i \leq l$. Ceux-ci sont donnés par la résolution du problème couplé constitué des équations (4.3) pour $1 \leq i \leq l$

$$\begin{cases} \dot{\mathsf{m}}_1 = \mathcal{M}_1(\dot{\mathsf{m}}_2, \ldots, \dot{\mathsf{m}}_l) \\ \quad \vdots \\ \dot{\mathsf{m}}_i = \mathcal{M}_i(\dot{\mathsf{m}}_1, \ldots, \dot{\mathsf{m}}_{i-1}, \dot{\mathsf{m}}_{i+1}, \ldots, \dot{\mathsf{m}}_l) \\ \quad \vdots \\ \dot{\mathsf{m}}_l = \mathcal{M}_l(\dot{\mathsf{m}}_1, \ldots, \dot{\mathsf{m}}_{l-1}). \end{cases} \tag{4.3}$$

4.2.2 Résolution du problème couplé

Sans perte de généralité et pour simplifier les notations, on se limitera à deux débits $\dot{\mathsf{m}}_1$ et $\dot{\mathsf{m}}_2$. On considère par la suite les équations et solveurs continus. Les mêmes calculs peuvent être effectués dans le cas discret auquel cas

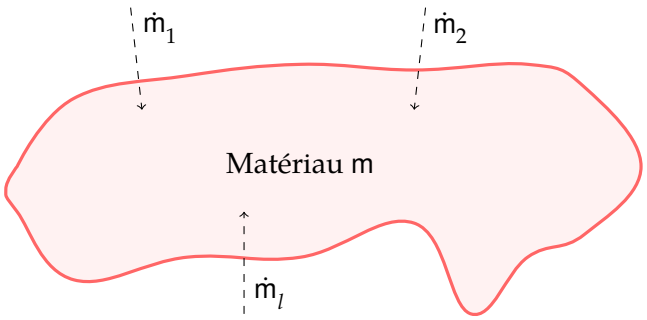


FIGURE 4.1 – Matériau m et débits ajoutés ou retirés $\dot{m}_i$ pour $1 \leq i \leq l$.

les solveurs et les intégrales utilisés ci-après sont discrétisés. L'équation d'évolution du matériau m (4.1) entre deux instants t^n et $t^{n+1} = t^n + \Delta t$ devient

$$m(t^{n+1}) = m(t^n) + \int_{t^n}^{t^{n+1}} \dot{m}_1(\tau)d\tau + \int_{t^n}^{t^{n+1}} \dot{m}_2(\tau)d\tau. \tag{4.4}$$

tandis que le problème couplé (4.3) devient

$$\begin{cases} \dot{m}_1 = \mathcal{M}_1(\dot{m}_2) \\ \dot{m}_2 = \mathcal{M}_2(\dot{m}_1). \end{cases} \tag{4.5}$$

Par la suite, le problème couplé (4.5) est résolu par un SCE et par un SCI. La résolution semi explicite du système par un SCE s'écrit

$$\begin{cases} \dot{m}_1^{n+1} = \mathcal{M}_1(\dot{m}_2^n) \\ \dot{m}_2^{n+1} = \mathcal{M}_2(\dot{m}_1^{n+1}). \end{cases} \tag{4.6}$$

Dans la première équation de (4.6), le solveur $\mathcal{M}_1$ *perçoit* une évolution du matériau m donnée par l'équation

$$m_{vu\ par\ 1}(t^{n+1}) = m(t^n) + \int_{t^n}^{t^{n+1}} \dot{m}_1^{n+1}(\tau)d\tau + \int_{t^n}^{t^{n+1}} \dot{m}_2^n(\tau)d\tau \tag{4.7}$$

tandis que dans la seconde équation de (4.6), l'évolution du matériau m *perçue* par le solveur $\mathcal{M}_2$ est donnée par

$$m(t^{n+1}) = m(t^n) + \int_{t^n}^{t^{n+1}} \dot{m}_1^{n+1}(\tau)d\tau + \int_{t^n}^{t^{n+1}} \dot{m}_2^{n+1}(\tau)d\tau \tag{4.8}$$

$$= m(t^n) + \int_{t^n}^{t^{n+1}} \dot{m}_1^{n+1}(\tau)d\tau + \int_{t^n}^{t^{n+1}} \mathcal{M}_2 \circ \mathcal{M}_1(\dot{m}_2^n)(\tau)d\tau. \tag{4.9}$$

Par différence des équations (4.7) et (4.9), il vient l'équation

$$\mathsf{m}_{\text{num.}} \overset{\text{def.}}{=} \mathsf{m}(t^{n+1}) - \mathsf{m}_{vu\ par\ 1}(t^{n+1}) = \int_{t^n}^{t^{n+1}} \left(\mathcal{M}_2 \circ \mathcal{M}_1(\dot{\mathsf{m}}_2^n) - \dot{\mathsf{m}}_2^n\right)(\tau)\mathrm{d}\tau. \tag{4.10}$$

correspondant à la création par le schéma explicite d'un matériau numérique $\mathsf{m}_{\text{num.}}$. On retrouve le résidu à l'interface donné par l'équation (2.11) chapitre 2.1 entre les solveurs 1 et 2 définit par

$$\mathcal{R}_{12}(\dot{\mathsf{m}}_2^n) = \mathcal{M}_2 \circ \mathcal{M}_1(\dot{\mathsf{m}}_2^n) - \dot{\mathsf{m}}_2^n. \tag{4.11}$$

Le matériau numérique créé par le schéma peut alors être donné en fonction de ce résidu à l'interface par

$$\mathsf{m}_{\text{num.}} = \int_{t^n}^{t^{n+1}} \mathcal{R}_{12}(\dot{\mathsf{m}}_2^n)(\tau)\mathrm{d}\tau. \tag{4.12}$$

En définissant une norme $\|.\|_{\text{mat}}$ sur les matériaux, par exemple la valeur absolue de sa masse, ce matériau numérique peut être majoré par

$$\|\mathsf{m}_{\text{num.}}\|_{\text{mat}} \leq \max_{\tau\in[t^n,t^{n+1}]} \left\|\mathcal{R}_{12}(\dot{\mathsf{m}}_2^n)(\tau)\right\|_{\text{mat}} \Delta t \tag{4.13}$$

permettant de retrouver que le schéma de couplage explicite donne toujours une erreur de l'ordre du macro pas de temps, voir [84] par exemple. Le matériau numérique créé par le schéma à chaque macro pas de temps est ensuite ajouté au matériau m décrit par son équation d'évolution (4.4) de sorte que la résolution est globalement conservative en masse. Néanmoins, comme on a pu le voir dans la section précédente 4.1, une *énergie numérique* se créée à chaque macro pas de temps et dans la plupart des cas l'énergie globale créée de l'instant initial à l'instant final ne se somme pas à zéro et peut devenir importante.

Naturellement, les mêmes calculs sont possibles avec une résolution du problème couplé (4.6) par un SCI. Dans ce cas, le matériau *numérique* créé par le schéma peut être donné en fonction de la valeur du débit $\dot{\mathsf{m}}_2^{n+1,\infty}$ calculé à convergence du schéma sur les itérations du débit $\dot{\mathsf{m}}_2^{n+1,k}$ par

$$\mathsf{m}_{\text{num.}} = \int_{t^n}^{t^{n+1}} \mathcal{R}_{12}(\dot{\mathsf{m}}_2^{n+1,\infty})(\tau)\mathrm{d}\tau. \tag{4.14}$$

Il peut être majoré en fonction de la tolérance $\epsilon_{\text{rel.}}$ du SCI par

$$\|\mathsf{m}_{\text{num.}}\|_{\text{mat}} \leq \max_{\tau\in[t^n,t^{n+1}]} \left\|\mathcal{R}_{12}(\dot{\mathsf{m}}_2^{n+1,\infty})(\tau)\right\|_{\text{mat}} \Delta t \tag{4.15}$$

$$\leq \epsilon_{\text{rel.}}\, \Delta t\,. \tag{4.16}$$

Ainsi, le matériau numérique peut être complètement contrôlé par la tolérance du SCI et rendu aussi petit que souhaité.

Les résultats précédents mettent bien en valeur l'impact *numérique* positif que peut avoir un SCI sur le problème de la vidange en comparaison des SCE. Cependant, ces résultats sont classiques pour ces schémas. On retrouve bien que

— l'erreur commise à chaque macro pas de temps par un SCE est de l'ordre du macro pas temps Δt dans l'équation (4.13).
— l'erreur commise à chaque macro pas de temps par un SCI est totalement contrôlée par la tolérance $\epsilon_{\text{rel.}}$ du schéma et peut donc être rendue la plus petite possible dans l'équation (4.13).

Pour ce problème, l'erreur numérique créée à chaque macro pas de temps mène à la création d'une énergie numérique globale de l'instant initial à l'instant final, comme dans la section précédente 4.1, qui peut être contrôlée par un SCI et qui peut devenir important pour un SCE.

Ce qu'il est montré par la suite est que l'utilisation de SCI peut aussi avoir un impact direct sur l'*implémentation logicielle* des solveurs dans la plate-forme, en particulier sur le *code* correspondant au solveur.

4.2.3 Incohérences physiques liées à la résolution explicite

Imaginons qu'il existe un instant $t^\star$ définit par

$$\int_{t^n}^{t^\star} \dot{\mathsf{m}}_2^{n+1}(\tau)\mathrm{d}\tau = -\left(\mathsf{m}(t^n) + \int_{t^n}^{t^\star} \dot{\mathsf{m}}_1^{n+1}(\tau)\mathrm{d}\tau\right) \tag{4.17}$$

auquel le matériau m disparaît. L'équation d'évolution (4.1) du matériau entre les instants t^n et t^{n+1} se simplifie en

$$\begin{aligned}
\mathsf{m}(t^{n+1}) &= \mathsf{m}(t^n) + \int_{t^n}^{t^{n+1}} \dot{\mathsf{m}}_1^{n+1}(\tau)\mathrm{d}\tau + \int_{t^n}^{t^{n+1}} \dot{\mathsf{m}}_2^{n+1}(\tau)\mathrm{d}\tau && (4.18)\\
&= \mathsf{m}(t^n) + \int_{t^n}^{t^{n+1}} \dot{\mathsf{m}}_1^{n+1}(\tau)\mathrm{d}\tau + \int_{t^n}^{t^\star} \dot{\mathsf{m}}_2^{n+1}(\tau)\mathrm{d}\tau && (4.19)\\
&= \int_{t^\star}^{t^{n+1}} \dot{\mathsf{m}}_1^{n+1}(\tau)\mathrm{d}\tau. && (4.20)
\end{aligned}$$

Lors de la résolution du couplage par un SCE, le solveur $\mathcal{M}_1$, résolu avant le solveur $\mathcal{M}_2$, est découplé de celui-ci et un retard en temps se créé. En particulier, le débit $\dot{\mathsf{m}}_1^{n+1}$, calculé à partir de $\dot{\mathsf{m}}_2^n$ dans (4.6), n'est pas nécessairement cohérent avec le débit $\dot{\mathsf{m}}_2^{n+1}$. Il est donc tout a fait possible qu'entre les instants $t^\star$ et t^{n+1} le débit $\dot{\mathsf{m}}_1^{n+1}$ corresponde à un débit sortant du matériau m, i.e.

$$\int_{t^\star}^{t^{n+1}} \dot{\mathsf{m}}_1^{n+1}(\tau)\mathrm{d}\tau \overset{\text{masse}}{\leq} 0. \tag{4.21}$$

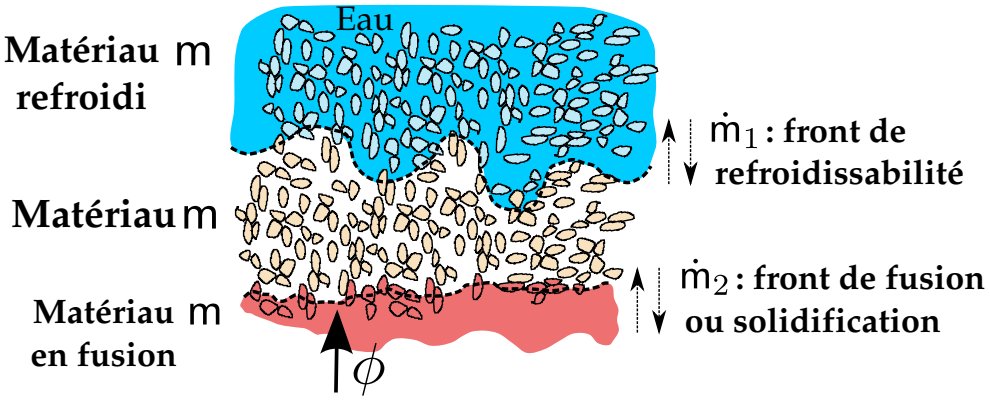


FIGURE 4.2 – Description du matériau m et des deux fronts *concurrents* de refroidissabilité et de fusion/solidification auxquels il est soumis.

alors que celui-ci *n'existe plus* à partir de l'instant $t^\star$ détecté par le solveur $\mathcal{M}_2$. Le débit sortant $\dot{m}_1^{n+1}$ a été surévalué par rapport au débit $\dot{m}_2^{n+1}$. La masse du matériau m à l'instant t^{n+1}, donnée par

$$m(t^{n+1}) = \int_{t^\star}^{t^{n+1}} \dot{m}_1^{n+1}(\tau)\mathrm{d}\tau \overset{\text{masse}}{\leq} 0, \tag{4.22}$$

devient négative et le matériau se trouve dans un *état physique incohérent*.

Ci-après est créé un cas de test et un scénario de vidange faisant apparaître par deux occasions la disparition du matériau m sous l'effet des deux débits $\dot{m}_1$ et $\dot{m}_2$. Ils sont ensuite utilisés dans la plate-forme PROCOR pour illustrer les propos tenus précédemment.

4.2.3.1 Cas de test et scénario de vidange

On considère le matériau poreux m avec une masse initiale de 50 kg (en marron dans la figure 4.2). Il est refroidi par une source d'eau et chauffé par un flux de chaleur ϕ. Le matériau m peut se trouver dans trois états : l'état *normal*, en marron dans la figure, l'état *refroidi*, en bleu dans la figure, ou l'état en *fusion*, en rouge dans la figure. On se concentre sur l'évolution du matériau dans son état normal.

L'évolution du matériau dans son état normal est gouvernée par le mouvement du front de refroidissabilité et de fusion/solidification. À ces deux fronts *concurrents* sont associés deux débits de masse $\dot{m}_1$ et $\dot{m}_2$ sortants ou entrants du matériau. Ces deux débits de masse correspondent à :

— une masse du matériau m dont une portion, en contact ou non avec l'eau, a été refroidie ou ne l'est plus correspondant respectivement à une avancée ou un recul du front de refroidissabilité. Le calcul de la

masse refroidissable est associé à l'évaluation *stationnaire* d'un flux critique maximal que l'eau en contact avec le matériau peut évacuer.
— une masse du matériau m dont une portion, sous l'effet ou non d'un flux de chaleur ϕ, rentre en fusion ou se solidifie correspondant respectivement à une avancée ou un recul du front de fusion/solidification. Le calcul de la masse en fusion est associé à la résolution d'un problème de Stefan à l'interface associée au front de fusion/solidification, c'est le même problème que celui décrit section 4.1.

Ces deux débits *concurrents* et *couplés* par l'état du matériau m sont calculés par deux solveurs $\mathcal{M}_1$ et $\mathcal{M}_2$ (associés respectivement à $\dot{m}_1$ et $\dot{m}_2$).

On a imaginé une séquence de mouvements des deux fronts menant à la vidange du matériau. Le scénario de vidange se divise en trois étapes :

1. À l'instant initial, une partie du matériau est refroidie tandis qu'une autre partie est chauffée par un flux de chaleur extérieur. Les deux débits $\dot{m}_1$ et $\dot{m}_2$ sont donc sortants. La surface refroidie du matériau ainsi que le flux de chaleur imposé sont tels que le matériau doit disparaître en moins d'un macro pas de temps. Ensuite, un macro pas de temps est encore calculé.
2. Après sa disparition, une partie du réfrigérant disparaît et le flux de chaleur imposé est moins important. Sous l'effet d'un débit entrant imposé constant et parce que le refroidissement et la fusion sont moins efficaces, le matériau m réapparaît. Deux macro pas de temps sont calculés.
3. Enfin, lors du dernier macro pas de temps, tout le réfrigérant réapparaît et le flux de chaleur imposé redevient important de telle sorte que tout le matériau m disparaît en moins d'un macro pas de temps.

Le macro pas de temps maximal utilisé pour les simulations est de $\Delta t = 100$ s. Il peut devenir plus petit si le schéma de couplage se synchronise sur l'événement de disparition du matériau pouvant survenir à l'intérieur du macro pas de temps, comme prévu dans le scénario.

4.2.3.2 Résultats numériques

La figure 4.3 donne l'évolution de la masse du matériau m suivant le scénario de vidange décrit précédemment. Lors du premier et du dernier Δt de calcul, entre les instants $t = 0$ s et $t = 100$ s et les instants $t = 400$ s et $t = 500$ s, le matériau disparaît sous l'effet des deux débits $\dot{m}_1^{n+1}$ et $\dot{m}_2^{n+1}$ concurrents. Le débit $\dot{m}_1^{n+1}$ n'étant pas correctement calculé par le SCE, la masse du matériau donnée par (4.22) devient négative. À $t = 100$ s, elle vaut $m \approx -17$ kg et vaut $m \approx -24$ kg à $t = 500$ s.

Comme prévu dans le scénario, l'événement de disparition du matériau est déclenché à l'intérieur du macro pas de temps. Cependant, tel qu'il est dans la plate-forme PROCOR, le SCE ne parvient pas à se synchroniser sur cet

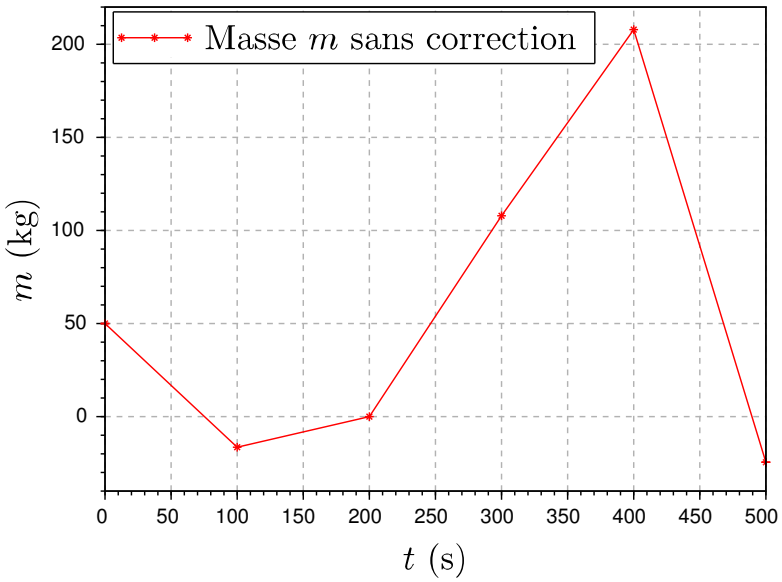


FIGURE 4.3 – Évolution de la masse du matériau calculée par un SCE *sans correction* dans le cas de masse négative.

événement. Les solveurs ne peuvent se synchroniser qu'à chaque macro pas de temps de couplage.

Pour résoudre au moins les problèmes de masse négative pour ce problème et de manière générale les incohérences physiques amenées par la résolution explicite, des correctifs sont utilisés dans la plate-forme PROCOR à la fin de chaque macro boucle en temps. Néanmoins, comme expliqué ci-après, ces correctifs ne sont pas des solutions viables et peuvent en plus être sources d'autres problèmes.

4.2.4 Solutions logicielles aux incohérences physiques amenées par la résolution explicite

De telles incohérences sont inhérentes aux SCE qui violent l'équilibre à l'interface entre les solveurs et les découplent sur la macro boucle en temps. Un décalage en temps se créé entre ceux-ci et en particulier entre les valeurs qu'ils calculent. Cependant, ces valeurs calculées lors de la résolution par le SCE, comme la masse du matériau, peuvent servir d'entrées à d'autres modèles. Si ces valeurs ne sont pas cohérentes, les autres modèles ne sont pas correctement initialisés. Il est donc nécessaire de corriger les incohérences créées par le schéma de couplage.

Dans les codes de calculs, la résolution de ces incohérences passent généralement par la création de portions de codes dédiées à la correction des situations non physiques causées par le SCE. Par exemple dans la plate-forme PROCOR, le pseudo code donné par l'algorithme 16 est dédié à la correction d'une masse négative à la fin d'un macro pas de temps d'un SCE. Il est exé-

Algorithme 16 Pseudo-code de correction d'une masse négative suite à la vidange d'un matériau dans PROCOR.

1: **if** $\mathsf{m}(t^{n+1}) \overset{\text{masse}}{\leq} 0$ **then**
2: $t^\star \leftarrow$ *instant de disparition du matériau* m
3: $m_{\leq 0} \overset{\text{masse}}{\leftarrow} \mathsf{m}(t^{n+1})$
4: $\dot{\mathsf{m}}_{1,\text{corr.}}^{n+1} \leftarrow -m_{\leq 0}/(t^{n+1} - t^\star)$ /* `débit de correction` */
5: $\dot{\mathsf{m}}_1^{n+1} \leftarrow \dot{\mathsf{m}}_1^{n+1} + \dot{\mathsf{m}}_{1,\text{corr.}}^{n+1}$ /* `correction du débit surévalué` */
6: $\mathsf{m} \overset{\text{masse}}{\leftarrow} 0$ /* `correction de la masse du matériau` */
7: **end if**

cuté à chaque fin de résolution sur un macro pas de temps du solveur $\mathcal{M}_2$. Lorsque celui-ci détecte une masse négative, ligne 1 de 16, un débit correctif rentrant $\dot{\mathsf{m}}_{1,\text{corr.}}^{n+1}$, donc positif, est créé ligne 5 puis ajouté ligne 6 au débit $\dot{\mathsf{m}}_1^{n+1}$ sortant négatif qui a été surévalué par le solveur $\mathcal{M}_1$. D'une part, le solveur $\mathcal{M}_2$ modifie des données calculées par un autre solveur. D'autre part, le code correctif dépend du schéma de couplage utilisé : si le solveur $\mathcal{M}_2$ avait été résolu avant le solveur $\mathcal{M}_1$, c'est le débit $\dot{\mathsf{m}}_2^{n+1}$ qui aurait du être corrigé.

Du retour d'expérience provenant de la plate-forme PROCOR, plusieurs problèmes se posent par rapport à ces corrections logicielles :

— ils obligent le modélisateur à résoudre des problèmes purement numériques ou le numéricien à faire des choix de modélisation physique. La séparation des métiers n'est plus respectée. Par exemple dans le premier cas, le modélisateur peut faire un mauvais choix numérique rendant les calculs instables car il n'a pas la maîtrise du schéma numérique ou dans le second cas, le numéricien peut faire un choix de modélisation discutable car il ne maîtrise pas les phénomènes physiques sous-jacent.

— les situations non physiques amenées par le schéma SCE sont souvent difficiles à détecter et, quand elles sont identifiées, résolues bien après l'étape de modélisation, généralement lors des calculs faits avec la plate-forme ou d'études industrielles, et donc bien souvent beaucoup trop tard.

Ces corrections prennent la forme d'un schéma prédicteur/correcteur. Cependant, elles doivent être créées au cas par cas. Elles peuvent potentiellement être sources de nouveaux problèmes physiques, numériques et/ou logiciels et il faut donc les éviter le plus possible.

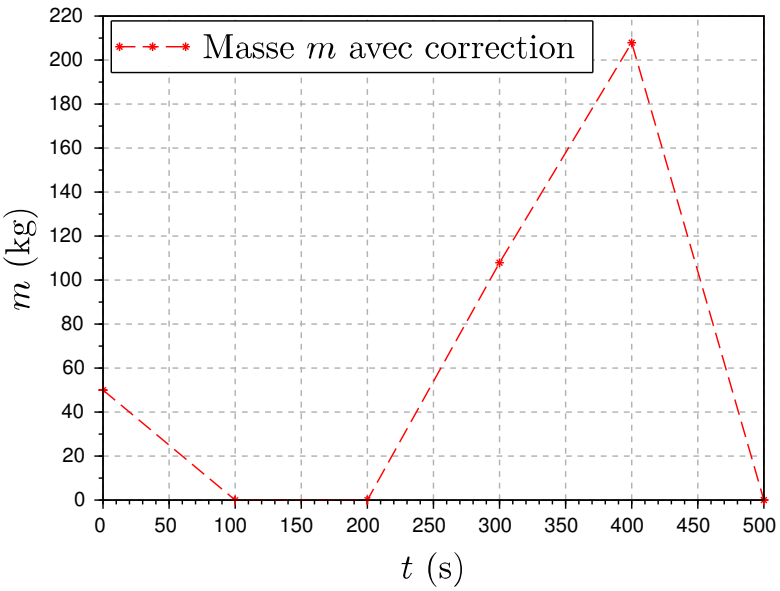


FIGURE 4.4 – Évolution de la masse du matériau calculée par un SCE *avec correction* dans le cas de masse négative.

4.2.4.1 Résultats numériques

Cette fois en utilisant le correctif donné par l'algorithme 16, la figure 4.4 donne l'évolution de la masse du matériau m suivant le scénario de vidange décrit précédemment. Pour le premier et le dernier Δt, lorsque la masse du matériau m était négative dans la figure 4.3 sans le correctif, la masse est bien ramenée à zéro par le correctif. Le débit sortant $\dot{\mathsf{m}}_1^{n+1}$ a été surévalué par le SCE et, pour le corriger, a été ensuite rehaussé du débit correctif entrant $\dot{\mathsf{m}}_{1,\text{corr.}}^{n+1}$ calculé à partir des masses négatives précédemment calculées. La figure 4.5 donne le débit sans correction $\dot{\mathsf{m}}_1^{n+1}$ et le débit avec correction $\dot{\mathsf{m}}_1^{n+1} + \dot{\mathsf{m}}_{1,\text{corr.}}^{n+1}$.

Comme attendu, les résultats numériques confirment le bon fonctionnement du correctif 16. Néanmoins, comme expliqué précédemment, cette solution n'est pas viable et ne devrait pas être utilisée. Ci-après, on explique comment la résolution implicite du problème couplé, permettant la synchronisation des modèles et un couplage fort entre ceux-ci, a pour conséquence la *suppression* de tels correctifs.

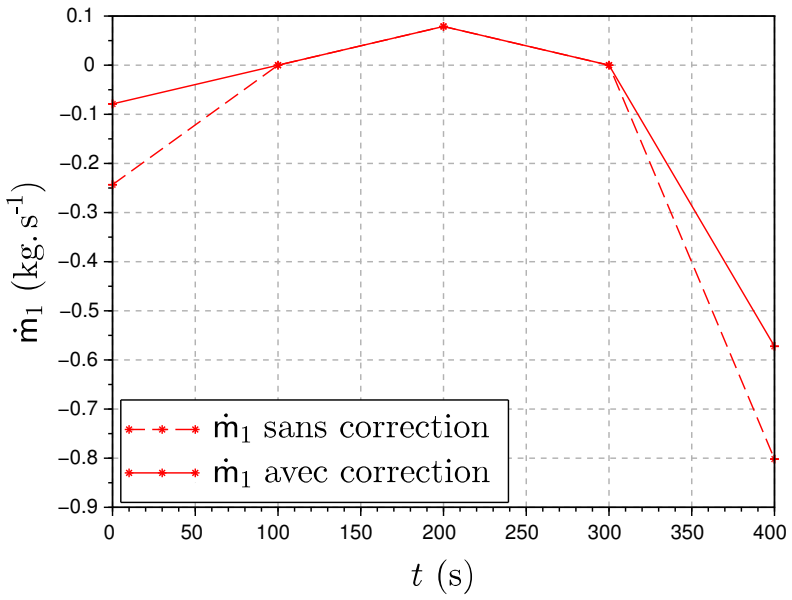


FIGURE 4.5 – Débits $\dot{m}_1$ avec et sans correction calculés lors de la vidange du matériau par un SCE.

4.2.5 Solutions aux incohérences physiques apportées par la résolution implicite

À convergence $k = \infty$ des itérations k d'un SCI sur un Δt durant la résolution du problème de couplage (4.3), l'équilibre à l'interface entre les modèles $\mathcal{M}_1$ et $\mathcal{M}_2$ est atteint avec une erreur contrôlée par la tolérance du schéma (voir (4.16)). Si de plus le schéma utilise l'algorithme de synchronisation détaillé section 2.2, il se synchronise *au cours des itérations* sur l'instant de disparition $t^\star$ de disparition du matériau. De ce fait, à convergence du schéma SCI synchronisé, le débit $\dot{m}_1^{n+1,\infty}$ est cohérent avec le débit $\dot{m}_2^{n+1,\infty}$ et le temps $t^{n+1,\infty}$ vaut $t^\star$ de telle sorte que

$$m(t^{n+1,\infty}) = m(t^n) + \int_{t^n}^{t^{n+1,\infty}} \dot{m}_1^{n+1,\infty}(\tau)\mathrm{d}\tau + \int_{t^n}^{t^{n+1,\infty}} \dot{m}_2^{n+1,\infty}(\tau)\mathrm{d}\tau \tag{4.23}$$

$$= m(t^n) + \int_{t^n}^{t^\star} \dot{m}_1^{n+1,\infty}(\tau)\mathrm{d}\tau + \int_{t^n}^{t^\star} \dot{m}_2^{n+1,\infty}(\tau)\mathrm{d}\tau \tag{4.24}$$

$$\overset{\text{masse}}{\approx} 0. \tag{4.25}$$

Un résidu correspondant à la convergence des itérations peut toujours être présent, d'où le $m(t^{n+1,\infty}) \approx 0$. Néanmoins, celui-ci peut être contrôlé par la tolérance $\epsilon_{\text{rel.}}$ du couplage aux interfaces entre modèles (voir (2.47) par

exemple) et par la tolérance $\epsilon_{\text{rel.}}^{\Delta t}$ (voir (2.48) par exemple) de l'algorithme de synchronisation.

Durant les itérations, il est possible que la masse du matériau devienne temporairement négative indiquant au schéma qu'il n'a pas encore convergé exactement sur les bons débits $\dot{\mathsf{m}}_1^{n+1}$ et $\dot{\mathsf{m}}_2^{n+1}$ et le bon temps de disparition $t^\star$. En conséquence, non seulement l'utilisation d'un SCI synchronisé rend *possible* la suppression des portions de code correctives évoquées précédemment telle que 16 mais, pour assurer la bonne convergence du schéma, rend la suppression *nécessaire*.

La vidange d'un matériau sous l'effet de plusieurs débits, sortants ou entrants, est un problème utilisé dans les modèles et applications industriels de la plate-forme. De ce fait, les solutions proposées précédemment pour la résolution du problème de couplage (4.5) peuvent directement être appliquées dans l'implémentation logicielle des modèles de la plate-forme. Pour chaque modèle calculant un débit pour la vidange d'un matériau, les portions de codes correcteurs tel que 16 ont été identifiées puis traitées. À présent, deux versions du même modèle cohabitent dans la plate-forme :

— une nouvelle version dans laquelle les codes correctifs ont été retirés. En enlevant ces sections de codes, cette version devient simplifiée et gagne en lisibilité, surtout pour le modélisateur physicien qui n'est pas nécessairement conscient des problèmes numériques liés au couplage. Cette version du modèle *doit être résolue avec un SCI muni de l'algorithme de synchronisation*. Si tel n'est pas le cas et si le modèle est résolu avec un SCE, des situations non physiques peuvent apparaître.
— l'ancienne version gardée pour des raisons de reproductibilité des résultats industriels, nécessairement résolue avec un SCE.

4.2.5.1 Résultats numériques

L'évolution globale de la masse m du matériau m est donnée par la figure 4.6. Elle est obtenue en résolvant le problème couplé (4.3) avec un SCI muni de l'algorithme de synchronisation. De ce fait, les versions sans codes correctifs des modèles sont utilisées. La courbe bleue correspond à la solution obtenue par un SCI muni de l'algorithme de synchronisation. La courbe noire correspond aux itérations du SCI jusqu'à convergence vers la courbe bleue. Lorsqu'il n'y a pas d'événement, par exemple durant l'étape 2 de l'instant $t \approx 138$ s à l'instant $t \approx 338$ s, le SCI converge rapidement en trois itérations à chaque Δt.

Lorsqu'il y a un événement, par exemple au premier ou au dernier Δt, la convergence peut être plus lente, respectivement 34 et 17 itérations. Néanmoins, la convergence peut être accélérée en modifiant les paramètres α et β de l'algorithme de synchronisation 5. Comme évoqué précédemment, durant les itérations détectant l'événement de disparition du matériau, le modèle peut rentrer dans un état physique incohérent, ici une masse négative,

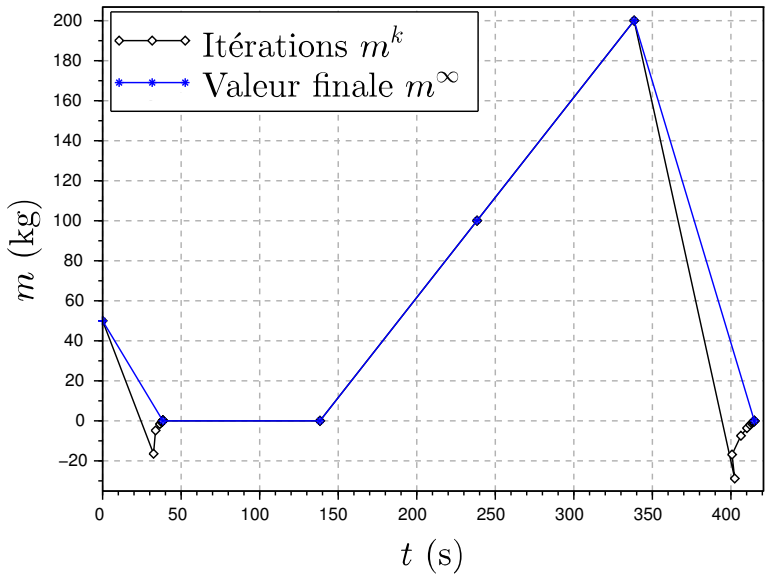


FIGURE 4.6 – Évolution de la masse m du matériau m donnée par un SCI muni de l'algorithme de synchronisation représentée par la courbe bleue et itérations de la masse m^k durant la résolution par le SCI jusqu'à convergence vers $m^\infty = m$ représentées par la courbe noire.

indiquant au schéma qu'il n'a pas encore convergé. Pour illustrer ces situations, un zoom sur la premier et le dernier Δt est donné dans les figures 4.7 et 4.8. Les itérations de la masse m^k sont négatives tout au long des itérations dans les deux cas et convergent vers $m^\infty \approx 0$, synonyme de disparition du matériau. Un résidu de masse existe toujours à la fin des itérations. Pour le premier le dernier Δt, le résidu m^∞ vaut respectivement -5×10^{-5} kg et 1.5×10^{-2} kg. Néanmoins, celui ci peut être contrôlé par les tolérances $\epsilon_{\text{rel.}}$ et $\epsilon_{\text{rel.}}^{\Delta t}$ du schéma.

4.2.6 Conclusion et remarques

Le but de cette section était *double*. D'une part, la résolution par différents des schémas explicites et implicites dans la plate-forme PROCOR du problème de vidange d'un matériau sous l'effet de plusieurs débits couplés permet :

— de remettre en valeur l'architecture de couplage créée et intégrée dans la plate-forme notamment par la ré-implémentation des modèles sous la nouvelle architecture et l'utilisation des nouveaux schémas de couplage.

— de remettre en valeur les bénéfices des SCI notamment en terme de

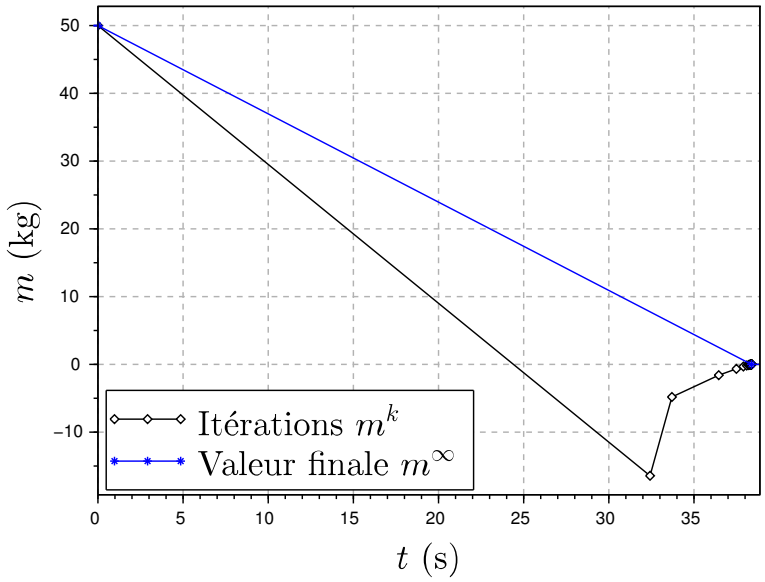


FIGURE 4.7 – Détection par le schéma de la disparition du matériau m lors du premier macro pas de temps.

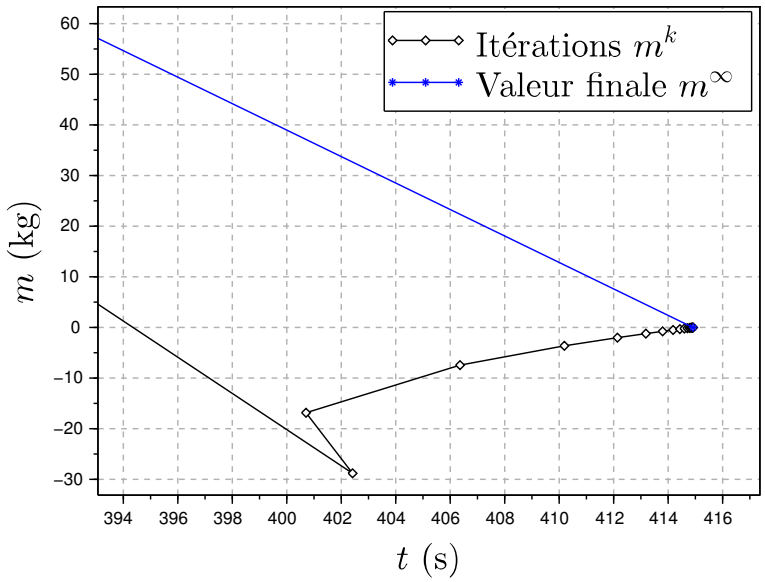


FIGURE 4.8 – Détection par le schéma de la disparition du matériau m lors du dernier macro pas de temps.

précision numérique, avec le couplage fort obtenu entre les modèles à la fin de chaque macro pas de temps, et en terme de synchronisation des modèles sur les événements majeurs tels que la disparition du matériau.

D'autre part, le but était aussi de montrer que l'utilisation de la nouvelle architecture peut aussi avoir un impact positif sur l'implémentation logicielle des modèles de la plate-forme. D'une part par la suppression des codes correctifs comme 16. Ces correctifs ne servent qu'à corriger les incohérences physiques inhérentes aux SCE et peuvent être sources d'erreurs numériques ou physiques. D'autre part, le ré-écriture des modèles et la création des schémas de couplage, qui étaient avant codés et fixés dans la plate-forme, permet de mettre clairement en valeur les couplages entre modèles et les échanges de données entre ceux-ci au cours de la résolution temporelle. Ainsi, de nombreuses fautes ont pu être découvertes, par exemple une quantité évalué avec un pas de temps de retard, être corrigées et les résultats des application industrielles servant pour les tests de non régression ont pu être mis à jour.

Aucune comparaison entre les solutions implicites et explicites et une solution de référence n'est donnée car une telle solution n'existe pas, ce qui est très souvent le cas dans le contexte AG. En général, une solution de référence peut être obtenue en utilisant un SCE qui a convergé en macro pas de temps, généralement pour des pas de temps très faibles mais inutilisables en pratique car beaucoup trop coûteux. Cependant, le modèle $\mathcal{M}_1$ utilisé pour calculé le débit $\dot{m}_1$ est un solveur stationnaire et l'utiliser pour des macro pas de temps trop faibles n'a pas de sens (ce modèle sera de nouveau utilisé dans la section suivante 4.3 et l'erreur créée lorsqu'il est utilisé hors de son domaine de validité sera illustrée). De ce fait, on ne peut juger de la qualité de la solution implicite par rapport à la solution explicite en terme de précision par rapport à une solution de référence. Néanmoins, la résolution implicite permet la synchronisation des modèles sur les événements majeurs, permet un couplage fort entre les modèles et permet de s'affranchir des contraintes de pas de temps imposées par les modèles (permettant ainsi d'utiliser des macro pas de temps suffisamment importants pour lesquels le schéma explicite est loin de son domaine de convergence). Toutefois, une solution de référence pourrait être obtenue par exemple en remplaçant *la modélisation stationnaire par une modélisation transitoire* ou par des *essais expérimentaux*.

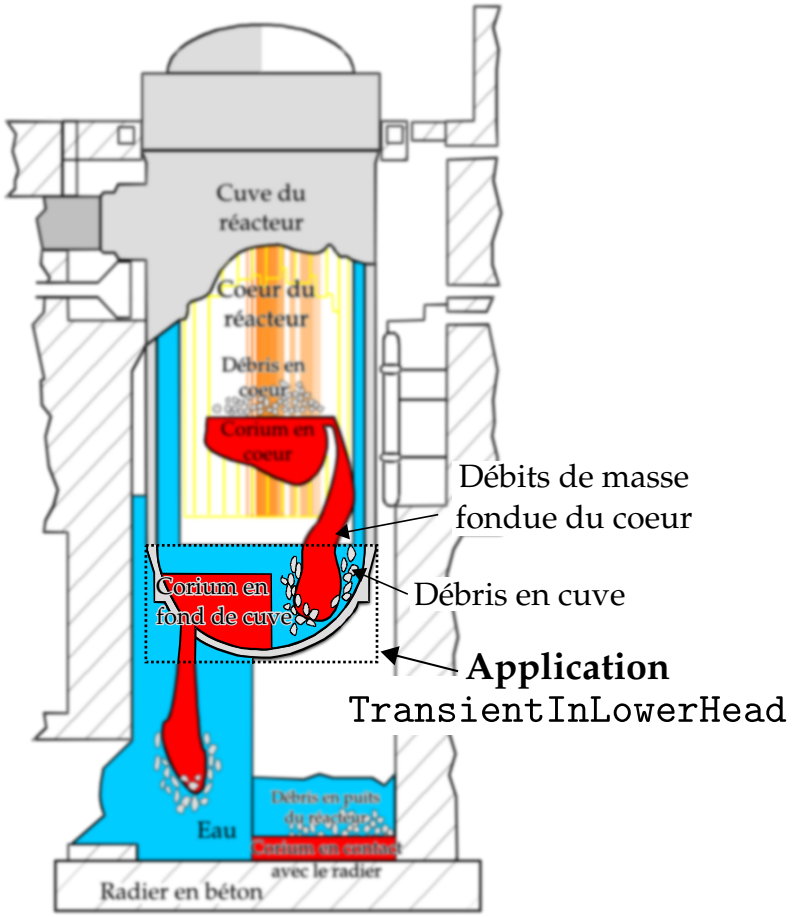


FIGURE 4.9 – L'application industrielle `TransientInLowerHead` modélise la partie non floutée de la propagation du corium dans le réacteur lors d'un accident grave, de la fonte du cœur du réacteur jusqu'à l'interaction entre le corium et le radier en béton.

4.3 Problème couplé d'une application industrielle PROCOR

4.3.1 Présentation de l'application

On présente dans cette section ce que modélise l'application industrielle `TransientInLowerHead` ainsi que les différents modèles de la plate-forme PROCOR constituant l'application.

Suite à la fonte du cœur du réacteur, un ou plusieurs bains de corium et un ou plusieurs lits de débris peuvent se trouver en fond de cuve du réacteur. Cette situation, correspondant à une partie de la phénoménologie complète de l'AG, est décrite dans la partie non floutée de la figure 4.9. L'application

industrielle PROCOR `TransientInLowerHead` modélise les interactions en fond de cuve entre un bain de corium, plusieurs lits de débris, la cuve, les structures environnantes et l'eau en et hors cuve. Dans l'application, le bain de corium et les lits de débits sont organisés en couches successives : un lit de débris inférieur en dessous d'un bain de corium stratifié en plusieurs couches de compositions différentes lui même en dessous d'un lit de débris supérieur. Ces différentes couches *interagissent* avec la cuve du réacteur, les structures internes de la cuve (non représentées dans la figure 4.9) et l'eau. Ci-après, on présente les différents modèles constituant l'application puis les couplages entre ces modèles.

4.3.1.1 Les modèles de l'application industrielle

L'application est constituée de plusieurs partitions correspondant à un partitionnement topologique du fond de la cuve du réacteur (voir figure 4.9). Ce partitionnement topologique correspond à une décomposition du fond de la cuve du réacteur en plusieurs sous domaines. Un modèle est associé à chaque partition. Chaque modèle peut lui même être composé de sous partitions. La division des partitions en sous partitions ne correspond pas toujours à un partitionnement topologique (ou à une décomposition du domaine en sous domaines). Elle peut aussi être liée à un décomposition de la physique du modèle en sous physique. Par exemple le modèle de bain de corium ou le modèle de lit de débris, présentés ci après. Au total, l'application est divisée en sept modèles : un modèle de fonte du cœur, un modèle de bain de corium stratifié, des modèles de lits de débris pour le lit au dessus du bain et le lit en dessous du bain, un modèle d'ablation de la cuve, un modèle d'ablation des structures internes et un modèle d'eau.

Le modèle de fonte du cœur. Il calcule des débits de masse fondue du cœur se relocalisant vers le bain de corium ou vers les lits de débris. Les données d'entrée concernant la fonte du cœur proviennent de calculs MAAP [33, 34] ou de coulées imposées par l'utilisateur et ne sont pas calculées par PROCOR dans cette application.

Le modèle de bain de corium stratifié. Le bain de corium est divisé en trois couches au maximum, une couche composée de métaux lourds en dessous d'une couche composée d'oxydes en dessous d'une couche composée de métaux légers. Ces trois couches sont entourées d'une croûte qui n'est actuellement modélisée qu'en terme de condition limite sur la température aux bords des couches. Au dessus de ces trois couches peut se trouver une couche d'acier, provenant par exemple de la cuve fondue par le bain. La formation de cette couche d'acier est une étape importante car, étant initialement fine, elle est associée à des flux de chaleur très importants imposés sur la cuve, c'est le phénomène de *Focusing Effect* (voir section 1.2). Au maximum, quatre couches

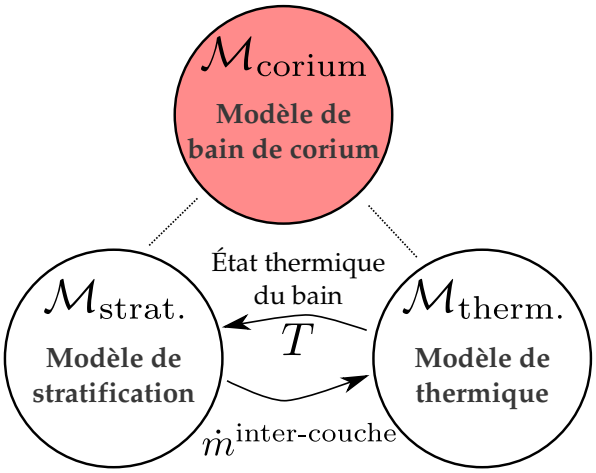


FIGURE 4.10 – Graphe de couplage des modèles constituant le modèle de bain de corium. Un trait en pointillés représente une relation modèle/sous-modèle.

peuvent composées le bain. Le modèle de bain de corium est lui même composé de deux modèles chaînés de manière explicite lors de la résolution du bain sur un macro pas de temps (ce chaînage est à l'heure actuelle encore fixé dans l'implémentation logicielle du modèle). À partir de la thermique du bain à un instant donné, un modèle de stratification *transitoire* calcule des débits de masse entre les différentes couches du bain. À partir de ces débits de masse, un modèle de thermique calcule les flux de chaleurs sortant des différentes couches du bain et imposés aux autres modèles. Ces deux modèles *couplés* constituant le modèle du bain de corium sont décrits par le graphe 4.10. Plus de détails sur le modèle de bain de corium et sur le phénomène de Focusing Effect potentiellement responsable du percement de la cuve peuvent être trouvés dans [68].

Le modèle de lit de débris. C'est ce modèle qui a été utilisé pour les résultats de la section précédente 4.2. Le modèle de lit de débris calcule, à partir d'un flux de chaleur imposé par exemple par le bain, un débit de masse en fusion du lit et un débit d'eau évaporée. Il est constitué de trois modèles, eux aussi chaînés de manière explicite dans l'implémentation logicielle du modèle. L'un des modèle calcule la puissance résiduelle $\dot{Q}$ du lit de débris. Un autre modèle calcule la quantité de débris refroidissable $m^{\text{ref.}}$ par l'eau environnante à partir d'un modèle stationnaire. Enfin, un autre modèle calcule la thermique de la partie du lit non refroidissable $m^{\text{non-ref.}}$, notamment le débit de masse fondue $\dot{m}^{\text{fus.}}$ sous l'effet du flux de chaleur reçu du bain. La partie non refroidissable du lit de débris est soumise à un problème de vidange décrite dans la section 4.2 par la figure 4.2. Les trois modèles *couplés* constituant le modèle de lit de

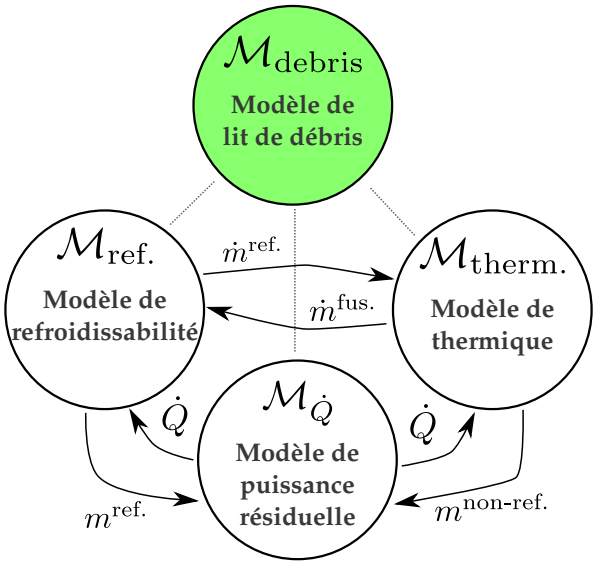


FIGURE 4.11 – Graphe de couplage des modèles constituant le modèle de lit de débris. Un trait en pointillés représente une relation modèle/sous-modèle.

débris sont décrits par le graphe 4.11.

Le modèle d'ablation de la cuve. Il calcule le débit de masse de cuve fondue sous l'effet du flux de chaleur reçu du bain de corium. Dans PROCOR, la cuve est maillée axialement. Dans chaque maille, la chaleur est transférée par conduction. Contrairement à la section 4.1, c'est ici un problème de Stefan à une phase (uniquement solide) qui est écrit.

Le modèle d'ablation des structures internes. Il calcule le débit de masse fondue des structures internes sous l'effet du flux de chaleur venant du bain.

Le modèle d'eau. À partir de la pression, sous l'hypothèse que l'on est à saturation, ce modèle décrit le bilan de masse en eau en et hors cuve.

Ces différents modèles sont couplés entre eux. Les principaux couplages sont détaillés dans la section suivante.

4.3.1.2 Les couplages entre modèles de l'application

Plusieurs couplages plus ou moins fort physiquement existent entre les modèles de l'application. Ci-après sont décrits les trois principaux couplages.

Un *fort couplage* en terme de flux de chaleur et de débits de masse existe entre les modèles de bain de corium, de lits de débris, d'ablation de la cuve et d'ablation des structures internes :

— Ayant une source interne de chaleur sous l'effet de la puissance résiduelle des éléments radioactifs, le bain de corium, initialement à une température très élevée ($\geq$ 2800 K), est le moteur de l'accident grave et fait fondre les lits de débris, la cuve du réacteur et ses structures internes.
— Des débits de masse provenant de la fonte des lits de débris, de l'ablation de la cuve et de l'ablation des structures internes sont alors ajoutés et relocalisés dans les différentes couches du bain de corium stratifié. Ces ajouts de masse peuvent modifier la stratification du bain, sa composition et sa température. Les flux de chaleur émis par le bain sont alors modifiés.

Ceci expliquant le couplage en terme de flux de chaleur et de débits de masse fondue entre les modèles de bain de corium, de lits de débris et d'ablation de la cuve et des structures internes.

Un autre couplage notoire existe entre le modèle de fonte du cœur et les modèles de bain de corium et de lits de débris :

— La relocalisation vers le bain de corium et les lits de débris des débits de masse fondue provenant du cœur, calculé par le modèle de fonte du cœur, dépend de la géométrie du bain, notamment de sa masse.
— La masse du bain de corium dépend naturellement des débits de masse reçus de la fonte du cœur et de la fonte des lits de débris.

Ceci expliquant le couplage en terme de masse du bain de corium et de débits de masse, du cœur et des lits de débris, entre les modèles de fonte du cœur, de bain de corium et de lits de débris.

Enfin, un couplage existe entre le modèle d'eau et les autres modèles :

— le volume et la localisation de l'eau présent en et hors cuve dépendent de l'état géométrique et thermique du bain de corium, des lits de débris, de la cuve et des structures internes qui peuvent être en contact avec l'eau. Par exemple, un lit de débris qui a été fortement refroidi peut faire évaporer une grande quantité d'eau.
— L'état géométrique et l'état thermique de ces différents éléments dépendent de la nature du contact avec l'eau. Par exemple, un lit de débris en contact avec l'eau sera refroidi et ne rentrera pas ou que faiblement en fusion et aucun ou un faible débit de masse sera envoyé sur le bain de corium.

Pour résumé, les différents modèles constituant l'application ainsi que les couplages entre modèles sont représentés dans le graphe de couplage 4.12. Dans la figure, on ne représente pas les sous modèles des modèles de bain de corium et de lits de débris. De ce fait, le graphe correspond au niveau le plus haut de l'arbre des modèles associé à l'application `TransientInLowerhead`. Dans le graphe, les arcs rouges représentent le couplage en flux de chaleur et débits de masse entre le bain de corium et les autres modèles. Les arcs noirs représentent le couplage en masse de bain et débits de masse fondue du cœur

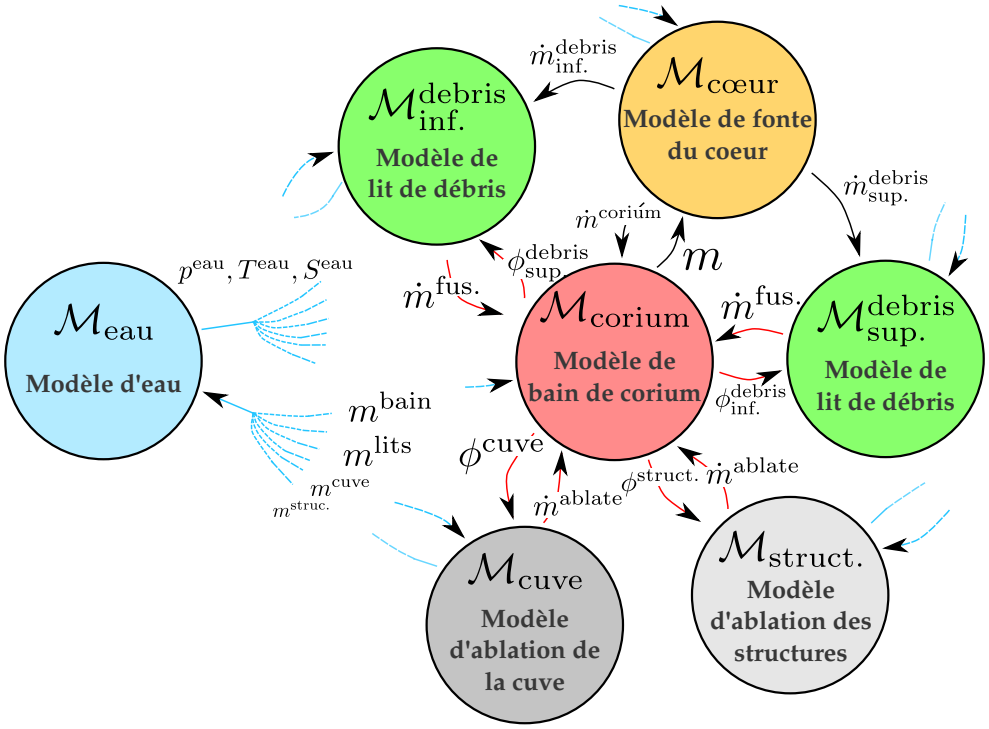


FIGURE 4.12 – Graphe de couplage *partiel* des modèles constituant l'application industrielle PROCOR `TransientInLowerHead`.

entre le modèle de fonte du cœur, le bain de corium et les lits de débris. Les arcs bleus représentent le couplage entre le modèle d'eau et les autres modèles. D'autres couplages, qui ne sont pas représentés dans le graphe, existent entre les modèles mais sont physiquement moins importants que ceux présentés ci-avant.

Dans les sections suivantes, on explique :

— L'implémentation logicielle d'*avant thèse* de l'application dans laquelle les modèles, couplages entre modèles et le schéma de résolution du couplage des modèles sont fixés.

— Le portage de l'application sous la nouvelle architecture et les bénéfices qu'apportent ce portage.

— L'implémentation logicielle d'*après thèse* de l'application permettant d'utiliser les différents schémas de couplage et la modularité offerte par l'architecture de couplage.

Cette présentation sera illustrée au fur et à mesure par des résultats numériques donnés par l'application sur

— un réacteur de Génération 3 (Gen3) avec un réflecteur lourd (voir section 1.2).

— un scénario d'accident correspondant à la perte du courant dans l'ins-

tallation nucléaire : "Loss Of Off Site Power" (LOOP) (voir section 1.1).

4.3.2 L'application PROCOR d'avant thèse

Les détails logiciels de l'application PROCOR que l'on donne ci-après permettent d'illustrer pourquoi les applications n'étaient pas adaptées au contexte AG.

Comme expliqué dans la présentation de l'architecture de la plate-forme PROCOR section 3.2, l'application hérite de la classe `TimeProcorApplication`. Les modèles la constituant sont donnés dans la méthode `constructModels`. L'ordre dans lequel les modèles sont résolus ainsi que les couplages entre les modèles sont ensuite *fixés* dans la méthode `apply`. L'application est ensuite résolue par macro pas de temps successifs de l'instant initial à l'instant final par l'algorithme 6 décrivant la résolution temporelle des applications PROCOR.

Le contexte AG est associé à une forte évolution des modèles constituant l'application. Dans l'implémentation logicielle d'avant thèse de l'application, le remplacement d'un modèle nécessite la reconstruction de toute l'application. De plus, la topologie du graphe de couplage n'est pas fixe au cours du transitoire de l'accident, des modèles peuvent apparaître ou disparaître au cours du temps suivant le scénario d'accident. Par conséquent, l'application doit être modulaire et doit pouvoir s'adapter au scénario d'accident choisi.

De plus, lors d'un AG, le transitoire peut devenir important, par exemple avec le phénomène de Focusing Effect responsable d'une ablation forte de la cuve. À chaque macro pas de temps, l'erreur de couplage est proportionnelle à Δt (voir [84] par exemple) et peut devenir conséquente et se propager tout au long de la simulation. La figure 4.13 représente la masse de cuve ablatée calculée par l'application avec un chaînage explicite des modèles pour différents macro pas de temps Δt lors du cas réacteur considéré [1]. Les solutions pour la masse ablatée convergent toutes vers une solution de référence donnée par un chaînage des modèles avec des macro pas de temps faibles, c'est à dire $\Delta t \leq 10$ s, pour lesquels le schéma explicite a convergé en temps pour cette quantité. Néanmoins, pour d'autres quantités calculées par l'application, des macro pas de temps aussi faibles ne sont pas envisageables dans le contexte physique et industriel des AG :

— D'une part pour des raisons de coût des calculs : l'application peut par exemple servir pour des calculs statistiques (PSA) nécessitant un nombre important de calculs.

— D'autre part, pour cette application, l'utilisation de macro pas de temps trop faible n'est pas cohérente physiquement avec la modélisation stationnaire utilisée dans certains modèles. La figure 4.14 donne la hauteur du lit de débris dans la cuve calculée en partie par un modèle

1. Réacteur Gen3 avec un scénario d'AG LOOP

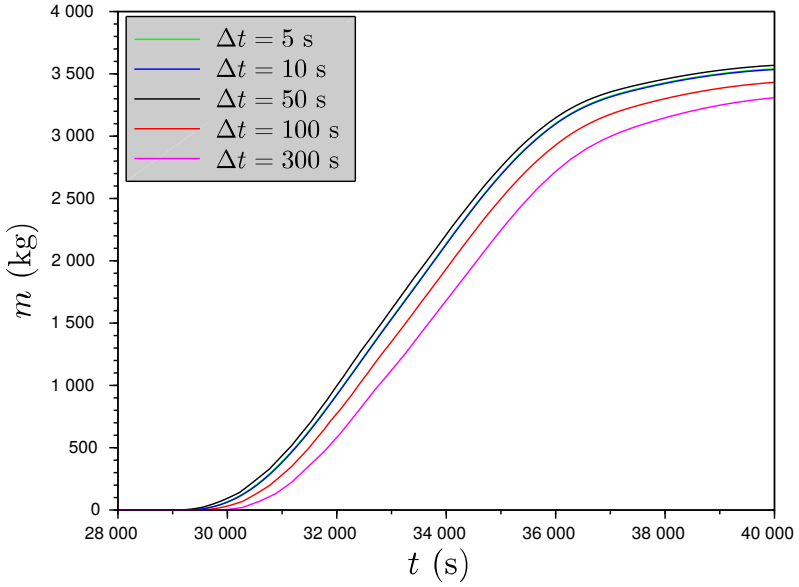


FIGURE 4.13 – Masse de cuve ablatée lors d'un cas réacteur calculée dans l'ancienne version de l'application industrielle PROCOR `TransientInLowerHead`.

stationnaire utilisé dans le modèle de lit de débris décrit 4.2. Pour des macro pas de temps faibles, ici $\Delta t \leq 10$ s, le modèle est utilisé hors de son domaine de validité physique et d'importants pics non physiques apparaissent.

Pour ces deux raisons, l'application est *contrainte* d'utiliser un macro pas de temps suffisamment important, c'est à dire $\Delta t \geq 50$ s. Dans PROCOR, on utilise généralement $\Delta t = 100$ s. Cependant, pour de tels macro pas de temps, l'erreur de couplage sur la masse totale ablatée de la cuve de l'instant initial à l'instant final, de 6 % pour $\Delta t = 100$ s et de 10 % pour $\Delta t = 300$ s, est trop importante.

Portée sous la nouvelle architecture de couplage, le graphe de l'application pourra être dynamique grâce à la modularité offerte par les différentes classes des modèles partitionnés (voir description de l'architecture section 3.3) et le couplage entre les modèles de l'application pourra être résolu même pour des macro pas de temps importants grâce aux SCI. Le portage de l'application et les résultats numériques obtenus sont détaillés dans les sections suivantes.

4.3.3 Portage de l'application sous la nouvelle architecture

L'application et les modèles la constituant ont été portés sous l'architecture de couplage décrite dans la section 3.3. Les modèles héritent de la classe `PartitionedModel`, leur permettant de déclarer leurs entrées, leurs sorties et

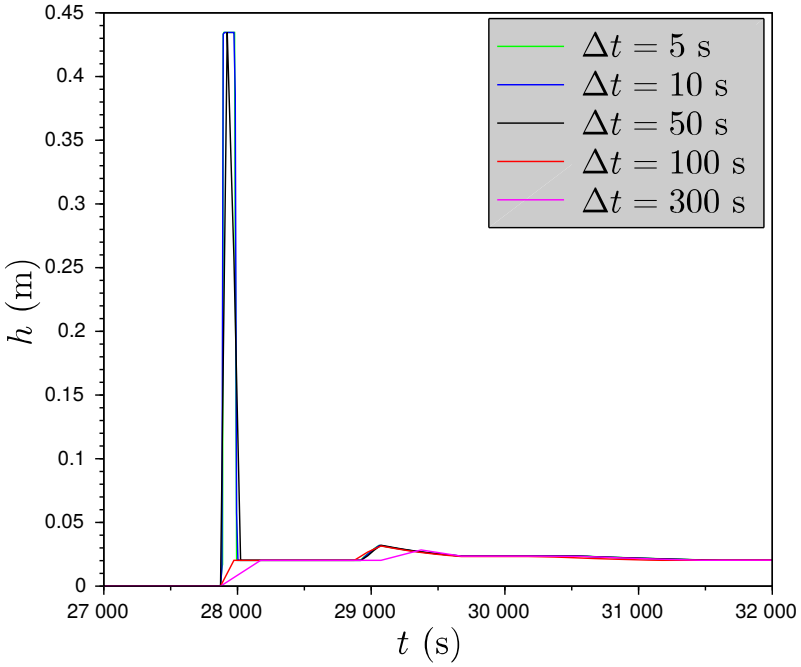


FIGURE 4.14 – Hauteur du lit de débris supérieur lors d'un cas réacteur calculée dans l'ancienne version de l'application industrielle PROCOR `TransientInLowerHead` avec un modèle stationnaire hors de son domaine de validité pour des macro pas de temps faibles ($\Delta t \leq 10$ s).

leurs partitions. À l'exception du modèle de bain de corium et du modèle de lit de débris, les modèles ne sont pas partitionnés. Ils sont donc des feuilles de l'arbre des modèles dont l'application est la racine. Le modèle de bain de corium est encore en cours de partitionnement et le couplage entre ses partitions est pour le moment toujours résolu de manière explicite. Le modèle de lit de débris est quant à lui totalement partitionné et peut être résolu de manière synchronisé avec n'importe quel schéma de couplage. C'est celui utilisé pour les résultats numériques du problème de la vidange dans la section 4.2.

Les communications entre modèles, avant écrites et fixées dans l'application, héritent maintenant de la classe `Communicator`. L'application est résolue en appelant sa méthode `solve` de la classe `PartitionedModel` avec une stratégie de couplage héritant de la classe `PartitionedStrategy` construite à partir des communicateurs.

Comme dans la section précédente 4.2, le portage de l'application et de ses modèles aura eu, en plus des bénéfices numériques apportés par l'utilisation d'un couplage implicite et synchronisé, un fort impact sur la qualité logicielle générale de l'application et sur la qualité des résultats industriels (corrections

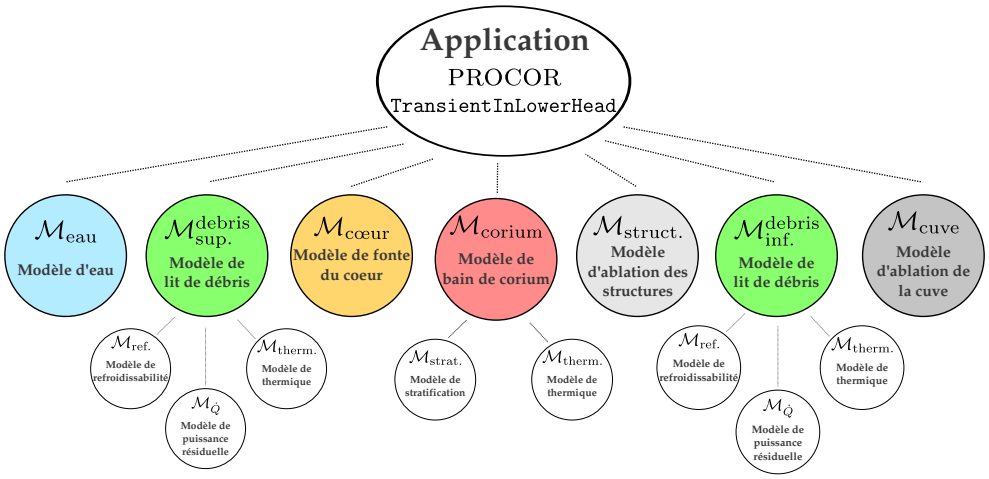


FIGURE 4.15 – L'arbre de modèles de l'application industrielle PROCOR `TransientInLowerHead` dans la nouvelle architecture.

de bugs et de problèmes liés à la transmission de données évaluées à de mauvais instants de la macro boucle en temps, meilleur séparation de métier entre le numéricien et le modélisateur, etc.).

4.3.4 L'application industrielle d'après thèse

L'application, telle qu'elle est implémentée dans l'architecture, est représentée par l'arbre de modèles 4.15 dont elle est la racine.

En héritant de la classe des modèles partitionnés `PartitionedModel`, les modèles rendent l'arbre de leurs sous modèles visible et accessible depuis l'extérieur. En particulier, chaque niveau de l'arbre est rendu accessible aux stratégies de couplage permettant ainsi aux couplages entre sous modèles du modèle père d'être résolus d'une manière spécifique définie dans la stratégie. La stratégie de couplage de l'application, constituée des stratégies de couplage des sous modèles et de leurs communicateurs, est représentée par le graphe de stratégies et de communicateurs figure 4.16.

Dans l'architecture, le couplage en flux de chaleur et débits de masse entre le bain de corium et les autres modèles est représenté par deux communicateurs, respectivement C^{ϕ} et $C^{\dot{m}}$ dans la figure 4.16. Comme décrit dans le diagramme UML 3.19, ils sont construits à partir d'un objet `Factory` auquel on a donné des objets `Projector`, représentant l'opération de projection faite dans la communicateur, des objets `Variable` et des objets `Parameter` représentant les variables d'entrées et de sorties des modèles. Par exemple, le projecteur du flux de chaleur est une fonction mathématique calculant, à partir d'un flux de chaleur par couches du bain, un flux de chaleur projeté sur les lits de débris,

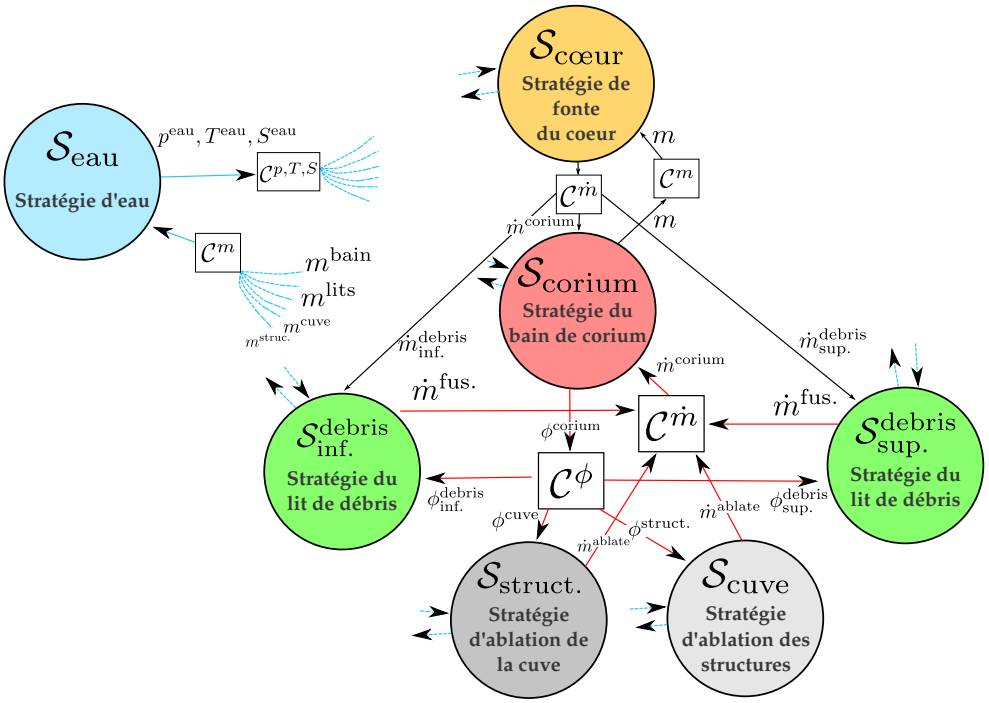


FIGURE 4.16 – L'arbre de modèles de l'application industrielle PROCOR `TransientInLowerHead` dans la nouvelle architecture.

la cuve et les structures internes. Les communicateurs ainsi construits sont capables des différentes opérations décrites dans le diagramme 3.19 comme la réception de données, l'envoi de données, la mesure du déséquilibre à l'interface entre les modèles ou un post-traitement telle que la relaxation des données à l'interface. Ces opérations rendent ainsi possibles différents schémas pour la résolution du couplage. Ils sont présentés ci-après.

4.3.5 Les différents graphes de séquençage des modèles de l'application

Deux stratégies pour la résolution du couplage des modèles de l'application sont actuellement utilisées :

— Une stratégie explicite utilisant un SCE et construite de manière à reproduire *exactement* les résultats obtenus avant le portage de l'application. Ainsi, on assure la non régression des résultats industriels.

— Une stratégie mixte explicite et implicite synchronisée permettant la résolution implicite de certains couplages forts physiquement et la résolution explicite d'autres couplages.

Comme expliqué dans la section 2.1, différents séquencements de modèles sur un macro pas de temps sont possibles pour la résolution explicite et im-

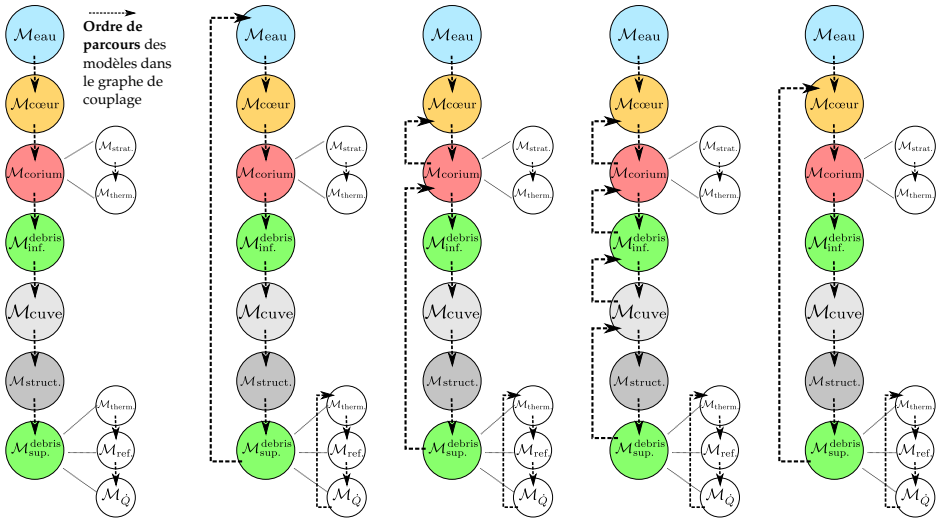


FIGURE 4.17 – Différents séquencements des modèles sur un macro pas de temps : pour une résolution explicite (*à gauche*) et des résolutions mixtes explicite/implicite (*ceux de droite*). La stratégie mixte la plus à droite est celle utilisée par la suite pour l'application.

plicite des couplages. Les différents graphes de séquencement choisis pour l'application sont donnés par la figure 4.17.

La stratégie explicite est construite à partir :

— des différentes stratégies des modèles de l'application et des communicateurs entre modèles (les stratégies et les communicateurs sont donnés dans la figure 4.16). De manière récursive, chaque modèle a sa propre stratégie de résolution du couplage de ses sous-modèles ;

— d'un chemin dans le graphe de couplage donné par le graphe de séquencement des modèles à gauche dans la figure 4.17. De même, récursivement, chaque modèle a son propre graphe de séquencement de ses sous-modèles.

La résolution de l'application avec cette stratégie explicite est ensuite faite en suivant l'algorithme de résolution explicite 11.

La stratégie utilisée pour une résolution plus précise du couplage de l'application est une stratégie mixte explicite/implicite. Différents calculs de l'application avec une résolution implicite de tous les couplage ont permis de mettre en évidence un couplage plus faible que les autres entre le modèle d'eau et les autres modèles. Grâce aux communicateurs aux interfaces entre le modèle d'eau et les autres modèles, les résidus de couplage $\mathcal{R}(\mathbf{b}^{n,k})$ ont pu être évalués permettant d'approcher la jacobienne des résidus $\propto \frac{\mathrm{d}\mathcal{R}}{\mathrm{d}\mathbf{b}}$. Cette jacobienne étant proportionnelle à la force du couplage entre les modèles, on a pu montrer numériquement que la force du couplage entre le modèle d'eau et

les autres modèles est plus faible que les autres. Ceci se traduisant notamment par un convergence très rapide vers zéro des résidus.

Par conséquent, le couplage entre le modèle d'eau et les autres modèles est résolu explicitement tandis que les autres couplages sont résolus implicitement. Naturellement, avec un tel schéma de résolution des couplages, à la fin de chaque macro pas de temps, un couplage fort est obtenu entre tous les modèles, sauf le modèle d'eau (le résidu de couplage à l'interface entre le modèle d'eau et les autres modèles reste suffisamment faible même avec une résolution explicite).

La résolution implicite et synchronisée entres les stratégies $\mathcal{S}_{\text{corium}}$, $\mathcal{S}^{\text{debris}}_{\text{inf.}}$, $\mathcal{S}^{\text{debris}}_{\text{sup.}}$, $\mathcal{S}_{\text{cuve}}$, $\mathcal{S}_{\text{struc.}}$ et $\mathcal{S}_{\text{cœur}}$ est faite grâce à l'algorithme de résolution implicite 12. De nouveau, différents séquencements des modèles sont possibles pour la résolution implicite des couplages sur un macro pas de temps. Ils sont donnés par les graphes de séquencement de droite dans la figure 4.17 dans lesquels une ou plusieurs boucles apparaissent. Par la suite, c'est le graphe de séquencement le plus à droite qui est utilisé. Néanmoins, ce choix est totalement arbitraire et mériterait d'être étudié plus en profondeur [2].

Avec cette stratégie explicite/implicite, il est ensuite possible de résoudre plus précisément le couplage des modèles de l'application. La figure 4.18 donne la masse ablatée totale calculée en résolvant l'application avec le nouvelle stratégie de couplage. Par rapport aux résultats précédents, donnés dans la figure 4.13, la masse ablatée donnée par la résolution explicite/implicite coïncide avec la solution de référence pour un macro pas de temps important de $\Delta t = 100$ s. Non représentées dans la figure, les solutions avec le schéma explicite/implicite pour d'autres macro pas de temps ($\Delta t \in \{10, 25, 100, 300\}$) coïncident aussi avec la solution référence. L'erreur relative par rapport à la référence sur l'état stationnaire atteint, c'est à dire la masse totale ablatée de cuve à la fin de la simulation, pour les différentes solutions, est donné dans le tableau 4.1. Dans le domaine d'utilisation de l'application, c'est à dire $\Delta t \geq 100$ s, le SCE est encore loin de son domaine de convergence en temps sur la masse ablatée de cuve : des erreurs relatives de l'ordre de 10 % sont commises par l'explicitation du couplage. À l'inverse, la résolution avec le schéma de couplage explicite/implicite est relativement indépendante de Δt, au détriment du temps de calcul si celui-ci est trop important (jusqu'à 15 itérations pour $\Delta t = 300$ s, sinon en moyenne 4 itérations pour $\Delta t = 100$ s). Ces résultats confirment que la résolution explicite/implicite du couplage permet de choisir relativement arbitrairement le pas de temps de couplage tout en obtenant des résultats précis et des calculs stables.

2. De manière général, pour un graphe de couplage quelconque, le meilleur choix de séquencement des modèles est fortement dépendent des couplages et doit être étudié au cas par cas. Le choix du meilleur séquencement des modèles ainsi que d'éventuelles solutions sont évoqués en ouverture de cette thèse.

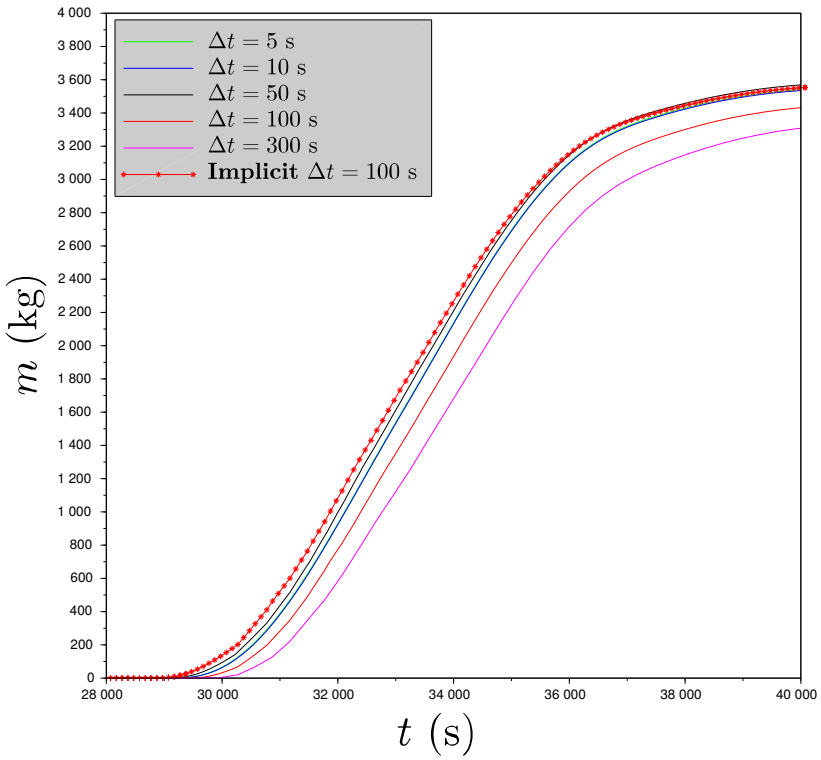


FIGURE 4.18 – Masse de cuve ablatée calculée par l'application industrielle PROCOR TransientInLowerHead pour un cas réacteur avec une résolution du couplage explicite pour différents Δt (courbes verte, bleue, noire, rouge et violette) et une résolution explicite/implicite (courbe rouge avec astérisques).

Δt (s)	5	10	25	100	300
$m^{\mathbf{exp}}_{\text{ablatee}}$ (kg)	3543	3540	3599	3337	3200
$\epsilon^{\mathbf{exp}}_{\text{rel}}$ (%)	0	0.08	1.6	6	10
$m^{\mathbf{imp}}_{\text{ablatee}}$ (kg)	3543	3541	3539	3555	3484
$\epsilon^{\mathbf{imp}}_{\text{rel}}$ (%)	0	0.05	0.11	0.33	1.6

TABLE 4.1 – Masse de cuve ablatée *totale* calculée par l'application industrielle PROCOR TransientInLowerHead pour un cas réacteur avec une résolution du couplage explicite et implicite pour différents Δt.

Ces résultats laissent entrevoir de bons espoirs d'une résolution précise et stable de l'application même pour des Δt importants imposés par les contraintes industrielles de temps de calcul ou des contraintes liées à la modélisation physique, par exemple l'utilisation de modèles stationnaires ne pouvant être calculés pour des macro pas de temps trop faibles.

4.3.6 Le couplage entre le bain de corium et la cuve

Dans les résultats précédents, l'ensemble du couplage donné par le graphe 4.12 a été considéré. Celui-ci a été résolu par les différentes stratégies de couplage et les bénéfices apportés par la stratégie explicite/implicite par rapport à la stratégie explicite ont été démontrés sur un cas réacteur et un scénario d'accident grave. Lors des différents tests amenant aux résultats, le couplage en flux de chaleur et débit de masse entre le bain de corium et la cuve s'est révélé être plus fort physiquement que les autres couplages. Par la suite, on y prête une attention particulière.

De plus, le couplage entre le bain de corium et la cuve et sa résolution précise est d'importance dans le cadre d'études sur la rétention du corium en cuve, notamment pour la stratégie IVR. Ce type d'études visent à montrer la bonne tenue de la cuve et en particulier le fait qu'elle ne se perce pas sous l'effet de la puissance thermique déposée par le bain sur sa paroi. De ce fait, le couplage doit être résolu précisément pour obtenir des résultats valides et utilisables pour les études. Par la suite, on se place dans le cadre de benchmarks proposés dans le contexte du projet européen H2020-IVMR pour le "In-Vessel Retention (IVR) tests codes benchmark" proposé par l'IRSN (voir [15] pour la définition du benchmark et [67] pour une description des cas tests utilisés dans la plate-forme PROCOR). Dans ce benchmark sont proposés différents cas correspondant à différentes configurations initiales du bain de corium. Ces différents cas permettent d'étudier l'impact de la stratification du bain de corium sur l'ablation de la cuve.

De ce fait, par la suite, on se concentre sur une configuration particulière de l'application industrielle `TransientInLowerHead` en ne considérant qu'un bain de corium en fond de cuve arrivé suite à la fonte du cœur, sans lits de débris, ni structures internes. Dans un premier temps, le couplage de l'application est détaillé, puis les deux cas extraits du benchmark sont décrits, enfin des résultats numériques sur ces cas seront donnés.

4.3.6.1 Détails sur le couplage

Le bain de corium et la cuve tels qu'ils sont modélisés dans l'application `TransientInLowerHead` de propagation du corium en fond de cuve sont décrits par la figure 4.19. Dans le modèle utilisé dans l'application, le bain de corium est stratifié en plusieurs couches. Une couche de métaux lourds, une

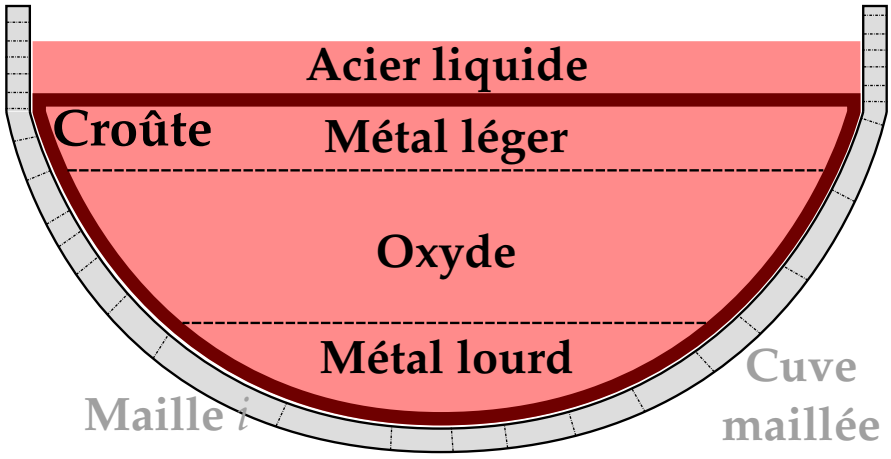


FIGURE 4.19 – Description du bain stratifié et de la cuve considérés dans l'application `TransientInLowerHead`.

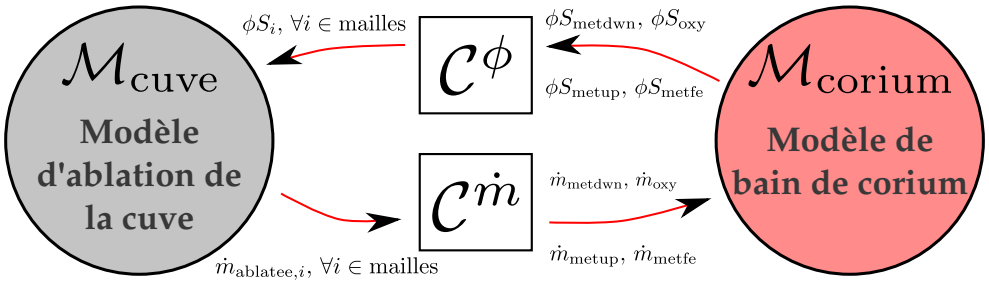


FIGURE 4.20 – Graphe de couplage simplifié de l'application industrielle PROCOR `TransientInLowerHead` dans le cadre de l'étude de l'interaction entre le bain de corium et la cuve.

couche d'oxydes et une couche de métaux légers sont entourées d'une croûte. Une couche d'acier liquide se trouve au dessus de cette croûte. Le bain de corium se trouve en fond de cuve. La puissance évacuée par les différentes couches du bain est imposée sur les mailles de la cuve et celle-ci est ablatée.

En ne considérant que le couplage entre le bain et la cuve, le graphe des stratégies et communicateurs de l'application est donné par la figure 4.20. Le communicateur C^{ϕ} permet au bain de corium de communiquer la puissance imposée sur la paroi de la cuve. Il reçoit en entrée les puissances

$$\phi S_{\text{metdwn}},\ \phi S_{\text{oxy}},\ \phi S_{\text{metup}},\ \phi S_{\text{metfe}}$$

évacuées des différentes couches du bain et calcule les puissances

$$\phi S_i,\ \forall i \in \text{mailles}$$

projetées sur les mailles radiales de la cuve. Le communicateur est, entre autre, composé d'un projecteur héritant, dans l'architecture, de la classe `FunctorModel`

(voir figure 3.19). De même, le communicateur $C^{\dot{m}}$ est composé d'un projecteur prenant en entrée des débits de masse ablatée pour chaque maille

$$\dot{m}_{\text{ablatee},i},\ \forall i \in \text{mailles}$$

et calculant des débits

$$\dot{m}_{\text{metdwn}},\ \dot{m}_{\text{oxy}},\ \dot{m}_{\text{metup}},\ \dot{m}_{\text{metfe}}$$

pour les différentes couches du bain de corium.

Remarques sur la manière de communiquer entre le modèle de bain et le modèle d'ablation de la cuve. Dans le couplage entre ces deux modèles figure 4.19, le modèle de bain est fortement sous-cyclé. Par exemple, pour les calculs, le micro pas de temps δt d'intégration en interne vaut 10^{-1} s, soit 1000 sous-cycles pour un macro pas de temps Δt de 100 s. De manière générale, on notera k le nombre de sous-cycles et on supposera que la longueur δt d'un sous cycle est constante au cours du macro pas de temps. Avant cette courte étude, le modèle de bain donnait au communicateur entre le modèle de bain et le modèle d'ablation de la cuve les puissances

$$\phi S_{\text{metdwn}},\ \phi S_{\text{oxy}},\ \phi S_{\text{metup}},\ \phi S_{\text{metfe}}$$

dernièrement évaluées par l'intégrateur en interne du modèle de bain. C'est à dire, entre les instants t^n et

$$t^{n+1} = t^n + k\delta t = t^n + \Delta t\,,$$

le modèle de bain évaluait les puissances totales par couche par

$$\phi S_{\text{couche}} = \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} \phi S_{\text{couche}}(\tau)\,\mathrm{d}\tau \approx \phi S_{\text{couche}}(t^{n+1})$$

et les transmettait au communicateur C^{ϕ}, revenant à faire une approximation du premier ordre (erreur en $O(\Delta t)$) sur l'intégrale. De ce fait, une erreur du premier ordre était commise par le modèle de bain. Le schéma implicite de couplage, permettant au maximum de retrouver les erreurs commises en interne des modèles, ne pouvait alors qu'être d'ordre 1 et ne pouvait faire mieux en terme de précision que le schéma de couplage explicite. Suivant les conseils donnés dans [83] pour le cas de modèles fortement sous-cyclés, l'évaluation de la puissance dans le modèle de bain a été modifiée et est maintenant don-

	Cas 1	Cas 2
m_{metfe} (kg)	6.3×10^4	
m_{metup} (kg)	0.0	
m_{oxy} (kg)	8.128×10^4	
m_{metdwn} (kg)	0.0	
$\dot{m}_{\text{oxy}}^{\text{metfe}}$ possible	non	oui

TABLE 4.2 – Configuration initiale des deux cas tests du benchmark extraits de [67].

née par

$$\begin{aligned}\phi S_{\text{couche}} &= \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} \phi S_{\text{couche}}(\tau)\,\mathrm{d}\tau \\ &\approx \frac{1}{\Delta t}\sum_{1\le j\le k} \phi S_{\text{couche}}(j\delta t)\delta t \\ &= \frac{1}{k}\sum_{1\le j\le k} \phi S_{\text{couche}}(j\delta t)\end{aligned}$$

permettant d'avoir une meilleur approximation de l'intégrale et donc transmettant moins d'erreur dans le couplage, donnant au schéma de couplage implicite un vrai intérêt.

Ci-après, des résultats extraits du benchmark calculés avec un résolution explicite du couplage dans l'application sont comparés à ceux donnés par la résolution implicite du couplage et à une solution de référence.

4.3.6.2 Description des cas tests

Deux cas tests sont extraits de [67]. Dans le premier cas test, on fait l'hypothèse qu'aucun transfert de masse n'est possible au travers de la croûte séparant la couche d'acier liquide et le bain oxyde. Dans le second cas, ce transfert devient possible. Dans les deux cas, le bain à l'intérieur de la croûte est initialement composé d'une couche oxyde. Les masses initiales de la couche d'acier liquide et de la couche oxyde sont données par le tableau 4.2. Pour le premier cas test, il n'y a pas d'interaction entre la couche d'acier liquide et la couche oxyde. Le seul aspect transitoire *notable* est l'épaississement de la couche d'acier liquide par l'ajout de masse ablatée de la cuve. Par conséquent, pour ce transitoire, la puissance déposée sur la cuve par le bain sera maximale lorsque la masse de la couche d'acier sera maximale. Pour le second test, le

transitoire est nettement différent. L'apport d'acier liquide modifie la composition de la couche oxyde. Le modèle de stratification du bain de corium (voir figure 4.10) prédit alors la formation d'une couche de métal léger au dessus de la couche oxyde. Les flux de chaleur sortant du bain, dépendant du type des couches présents dans le bain, sont alors modifiés. De plus, l'amincissement de la couche d'acier liquide peut engendrer des flux de chaleur importants (phénomène de Focusing Effect).

Décrite dans la définition du benchmark [15], la cuve utilisée [3] est une cuve hémisphérique d'un rayon de 2 m avec une partie supérieur cylindrique d'une hauteur de 1 m. Son épaisseur est de 15 cm. Pour l'étude, elle est maillée radialement avec 20 mailles.

Contrairement au cas étudié précédemment, aucun modèle stationnaire n'est utilisé dans la modélisation pour les deux cas tests. La résolution du couplage avec un macro pas de temps faible est donc possible et permet d'obtenir une solution de référence. Elle est donnée par la résolution explicite du couplage avec un macro pas de temps Δt de 5 s pour lequel l'application a convergé en pas de temps.

Le couplage entre le bain de corium et la cuve est résolu dans l'application de manière explicite et implicite pour différents macro pas de temps $\Delta t \in \{10, 20, 50, 100, 200\}$ de couplage. Par exemple, pour les calculs donnés dans le cadre du benchmark, Δt vaut 100 s. On compare, avec la solution de référence, au cours du transitoire, différentes valeurs d'intérêt dans le cadre de la stratégie IVR de rétention du corium en cuve :

— la masse m_{ablatee} ablatée de cuve.
— la puissance ϕS_i déposée sur les mailles i de la cuve par les différentes couches du bain de corium.
— l'épaisseur e de la cuve.

Pour mesurer l'erreur relative sur une quantité q commise entre une résolution explicite ou implicite par rapport à la solution de référence, on utilisera l'opérateur $\epsilon^{\text{rel.}}$ défini par

$$\epsilon^{\text{rel.}}(q^{\mathbf{sol.}}, \|.\|) = \frac{\left\|q^{\text{ref.}} - q^{\text{sol.}}\right\|}{\|q^{\text{ref.}}\|} \quad \text{pour sol.} \in \{\text{exp.}, \text{imp.}\}. \tag{4.26}$$

4.3.6.3 Résultats numériques pour le premier cas test

La figure 4.21 donne la masse ablatée de la cuve pour une résolution explicite (en rouge) et une résolution implicite (en bleue) du couplage par rapport à la solution de référence (en noire). Comme attendu, toutes les solutions implicites sont confondues avec la solution de référence (seule la solution pour $\Delta t = 100$ s est représentée dans la figure). Pour les différents Δt, l'erreur rela-

3. Cuve similaire à celle utilisée pour la conception du réacteur AP1000

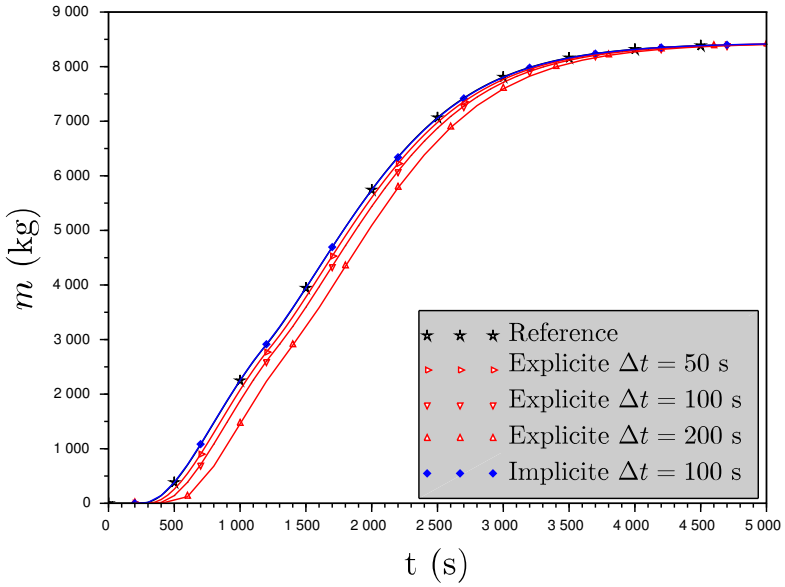
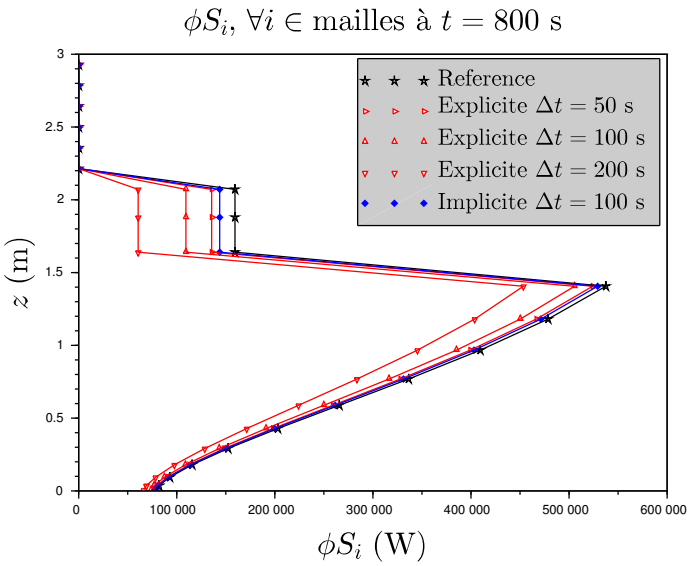


FIGURE 4.21 – **Premier cas test** : comparaison de la masse de cuve ablatée entre la solution de référence et des solutions explicites et implicites.

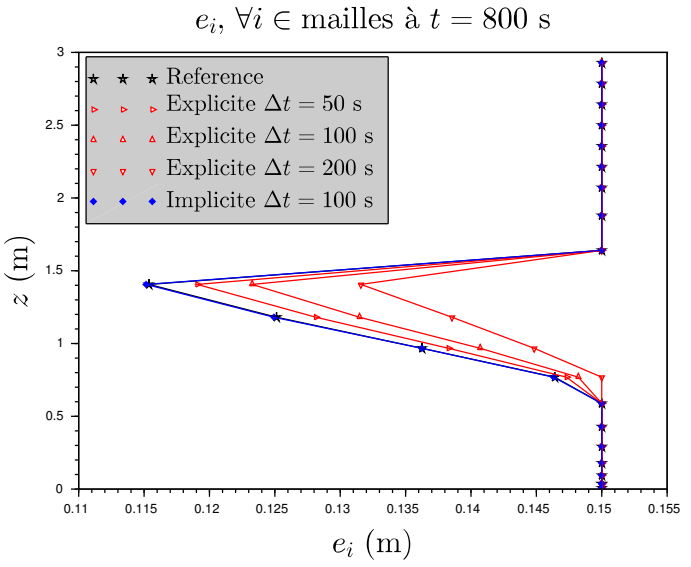
Δt	10	20	50 s	100 s	200 s
$\epsilon^{\text{rel.}}(m^{\textbf{exp.}}_{\text{ablatee}}, \|.\|_{\text{inf.}})$ (%)	0.5	1.12	2.23	4.62	9.44
$\epsilon^{\text{rel.}}(m^{\textbf{imp.}}_{\text{ablatee}}, \|.\|_{\text{inf.}})$ (%)	0.01	0.12	0.2	0.12	0.33

TABLE 4.3 – **Premier cas test** : erreur relative sur la masse de cuve ablatée entre la solution de référence et les solutions explicites et implicites pour différents Δt.

tive en norme infinie sur la masse ablatée de cuve est donnée dans le tableau 4.3. La résolution implicite est relativement indépendante du macro pas de temps de couplage Δt. La convergence en temps du schéma de couplage explicite, symbolisée par la double barre dans la tableau, est effective pour $\Delta t \leq 10$ s. Pour $\Delta t \geq 100$ s, des erreurs relatives de près de 5 % et 10 % sont commises. La masse ablatée de cuve vient s'ajouter à la masse m_{metfe} de la couche d'acier liquide et augmente la surface d'échange entre celle-ci et la cuve. La puissance déposée par la couche d'acier liquide sur la cuve est d'autant plus grande que la surface d'échange est importante. La figure 4.22a donne la puissance ϕS_i déposée sur chaque maille i de la cuve à l'instant $t = 800$ s du transitoire et la figure 4.22b donne l'épaisseur de chaque maille. La résolution explicite (solutions rouges dans 4.22a) du couplage a tendance a sous évaluer l'abla-



(a)



(b)

FIGURE 4.22 – **Premier cas test** : à $t = 800$ s, puissances déposées sur chaque maille i par le bain de corium sur la cuve (*haut*) et épaisseurs de chaque maille i de la cuve (*bas*).

Δt	10	20	50 s	100 s	200 s
$\epsilon^{\text{rel.}}(e_i^{\textbf{exp.}}, \|.\|_{\text{inf.}})$ (%)	0.7	1.8	2.46	5.26	10.8
$\epsilon^{\text{rel.}}(e_i^{\textbf{imp.}}, \|.\|_{\text{inf.}})$ (%)	0.01	0.07	0.09	0.14	0.21

TABLE 4.4 – **Premier cas test** : erreur relative sur l'épaisseur de la cuve entre la solution de référence et les solutions explicites et implicites pour différents Δt.

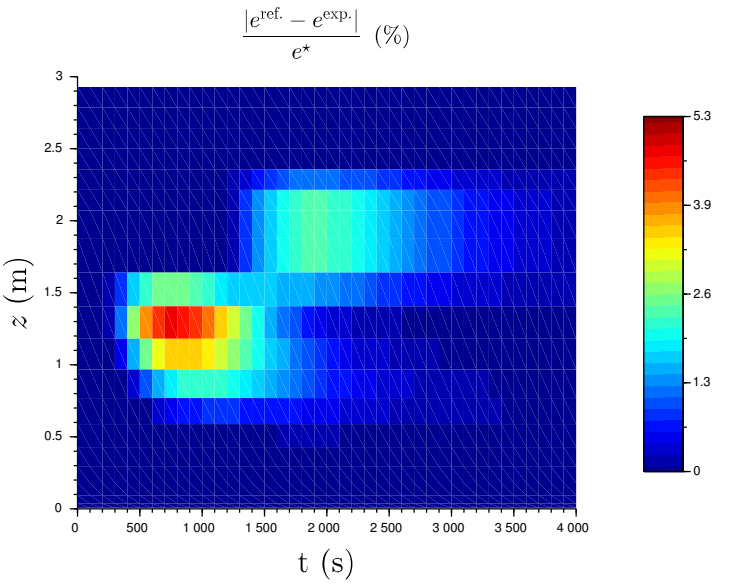
tion de la cuve, donc la surface d'échange et donc la puissance imposée sur la cuve. L'épaisseur de la cuve pour la résolution explicite (solutions rouges dans 4.22b) est donc sur évaluée par rapport à la solution de référence. La résolution implicite permet d'évaluer correctement la surface d'échange et la puissance déposée par le bain sur la cuve (solution bleue dans 4.22a). L'épaisseur de la cuve donnée par la résolution implicite (solution bleue dans 4.22b) coïncide avec la solution de référence. Il en est de même pour la résolution implicite avec d'autres macro pas de temps, ce qui est confirmé dans le tableau 4.4 donnant l'erreur relative à l'instant $t = 800$ s sur l'épaisseur de la cuve.

Pour conclure sur ce premier cas test, les écarts en valeur absolu au cours du temps entre les solutions de référence et explicite et les solutions de référence et implicite (*bas*) pour $\Delta t = 100$ s sur l'épaisseur de la cuve (écart normalisé par rapport à l'épaisseur initiale $e^\star = 15$ cm de la cuve) sont donnés respectivement par les figures 4.23a et 4.23b. Pour la résolution explicite et la résolution implicite du couplage, l'erreur maximale est commise au début du transitoire. Elle vaut plus de 5 % pour la résolution explicite et reste inférieure, tout au long de la simulation, à 0.15 % pour la résolution implicite.

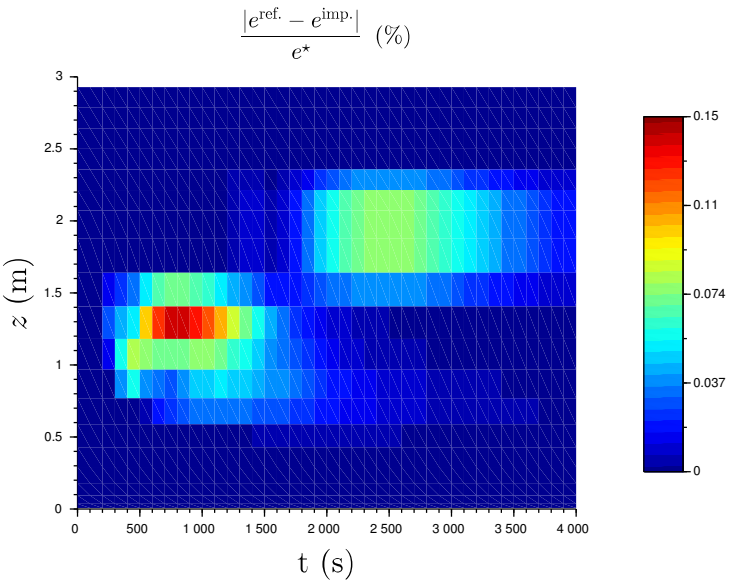
4.3.6.4 Résultats numériques pour le second cas test

Confirmé dans la figure 4.24, une couche de métal léger apparaît au dessus de la couche oxyde par transfert de masse au travers de la croûte. Par conséquent, la couche d'acier liquide au dessus de la croûte s'amincie et peut devenir très fine à la fin du transitoire. Ceci peut engendrer des phénomènes de Focusing Effect et le percement de la cuve. Le transitoire de stratification et les couches présentes dans le bain étant différentes, les puissances déposées par les couches sur la cuve sont différentes du cas précédent et *le transitoire de la masse ablatée est modifiée*.

La figure 4.25 donne la masse ablatée de la cuve pour la résolution explicite (en rouge) et pour la résolution implicite (en bleue) du couplage par rapport à la solution de référence (en noire). Pour une première partie du transitoire, $t \leq 1700$ s, la résolution implicite coïncide avec la référence. Pour la seconde partie du transitoire, $t > 1700$ s, la résolution implicite s'éloigne de la



(a)



(b)

FIGURE 4.23 – **Premier cas test** : écart en valeur absolu au cours du temps entre les solutions de référence et explicite (*haut*) et les solutions de référence et implicite (*bas*) pour $\Delta t = 100$ s sur l'épaisseur de la cuve (écart normalisé par rapport à l'épaisseur initiale $e^\star = 15$ cm de la cuve).

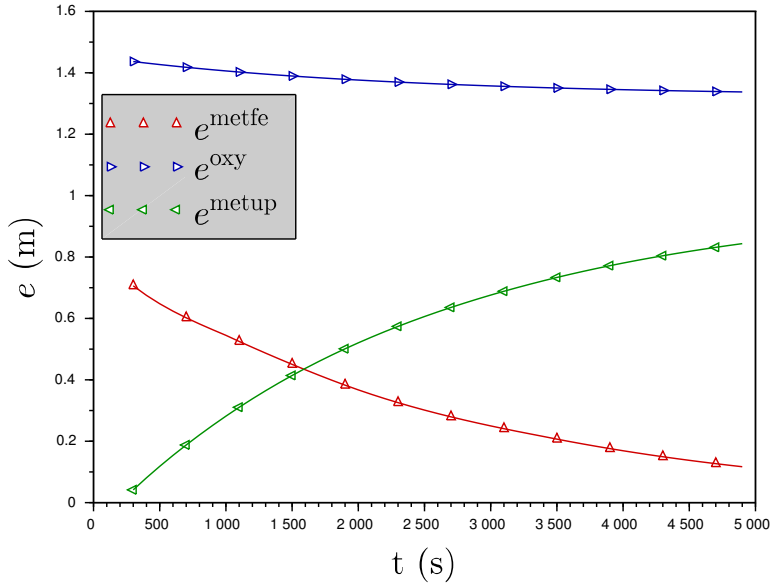


FIGURE 4.24 – **Second cas test** : épaisseurs des couches du bain données par la solution de référence.

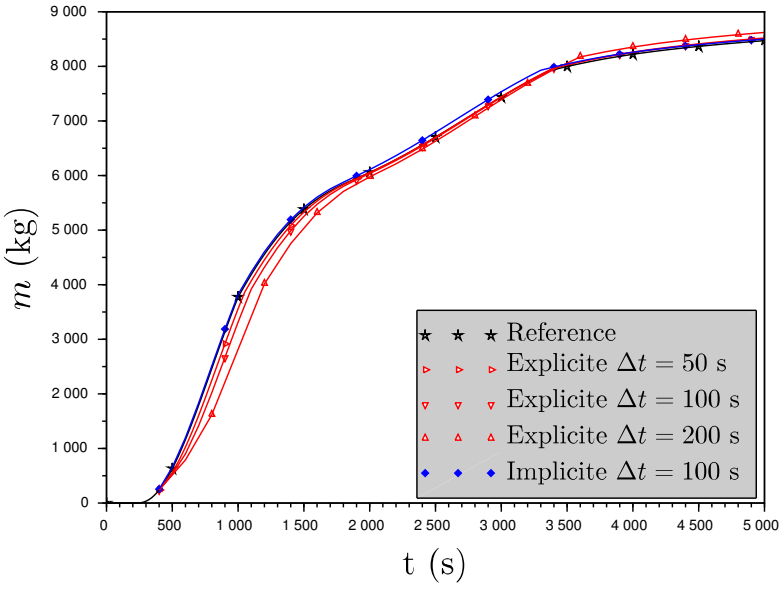


FIGURE 4.25 – **Second cas test** : comparaison de la masse de cuve ablatée entre la solution de référence et des solutions explicites et implicites.

Δt	10	20	50 s	100 s	200 s
$\epsilon^{\text{rel.}}(m^{\textbf{exp.}}_{\text{ablatee}}, \|.\|_{\text{inf.}})$ (%)	0.42	1.28	2.62	5.53	11.31
$\epsilon^{\text{rel.}}(m^{\textbf{imp.}}_{\text{ablatee}}, \|.\|_{\text{inf.}})$ (%)	0.02	0.4	0.6	1.15	1.92

TABLE 4.5 – **Second cas test** : erreur relative sur la masse de cuve ablatée entre la solution de référence et les solutions explicite et implicite pour différents macro pas de temps.

Δt	10	20	50 s	100 s	200 s
$\epsilon^{\text{rel.}}(\dot{Q}^{\textbf{exp.}}_{\text{oxy.}}, \|.\|_{\text{inf.}})$ (%)	1.4	3.3	7.06	15.17	30.47
$\epsilon^{\text{rel.}}(\dot{Q}^{\textbf{imp.}}_{\text{oxy.}}, \|.\|_{\text{inf.}})$ (%)	1.3	3.2	7.07	15.16	30.46

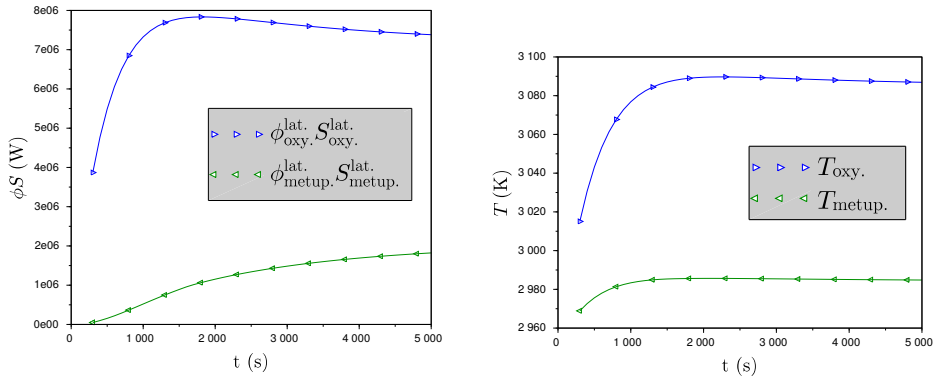
TABLE 4.6 – **Second cas test** : erreur relative sur la puissance résiduelle de la couche oxyde du bain entre la solution de référence et les solutions explicites et implicites pour différents Δt.

solution de référence. Ceci est confirmé dans le tableau 4.5 donnant l'erreur relative sur la masse ablatée pour les résolutions explicite et implicite pour les différents Δt. Pour $\Delta t = 100$ s, l'erreur relative pour une résolution implicite est de 1.15 % alors que pour le premier cas test, elle était 0.14 % (voir tableau 4.3). Elle est ici presque dix fois supérieure. L'erreur maximale est toujours commise dans la second partie du transitoire, pour $t > 1700$ s.

Cette erreur est créée dans le modèle de bain de corium sur l'évaluation de la puissance résiduelle des couches oxyde et métal léger. En effet, les partitions de stratification et de thermique dans le modèle de bain (voir figure 4.10) sont chaînées :

1. le modèle de stratification calcule, entre autres, la masse et la composition des différentes couches du bain à partir de la puissance résiduelle du bain répartie sur les différentes couches.
2. le modèle de thermique, entre autres, répartie la puissance résiduelle du bain en fonction de la masse et de la composition des différentes couches.

Le chaînage explicite de ces deux partitions *couplées* créé une erreur de l'ordre de Δt. Ceci est confirmé dans le tableau 4.6 donnant l'erreur relative entre la puissance résiduelle $\dot{Q}_{\text{oxy.}}$ de la couche oxyde du bain calculée par la solution de référence et les solutions explicites et implicites. Pour le même Δt, l'erreur sur la puissance résiduelle pour la résolution implicite, qui peut être importante (plus de 15 % pour $\Delta t = 100$ s), est *toujours la même* que celle commise lors de la résolution explicite. Le schéma implicite de *couplage* ne peut corriger l'erreur en *interne* commise dans le modèle de bain.



(a) (b)

FIGURE 4.26 – **Second cas test** : puissances latérales (*droite*) et températures moyennes (*gauche*) des couches oxyde (oxy.) et métal léger (metup.) du bain.

L'erreur sur l'évaluation des puissances résiduelles $\dot{Q}_{\text{oxy.}}$ et $\dot{Q}_{\text{metup.}}$ se propage dans l'évaluation des puissances $\phi^{\text{lat.}}_{\text{oxy.}} S^{\text{lat.}}_{\text{oxy.}}$ et $\phi^{\text{lat.}}_{\text{metup.}} S^{\text{lat.}}_{\text{metup.}}$ imposées par le bain sur la cuve puis dans le couplage (on se retrouve dans une situation similaire corrigée section 4.3.6.1 lorsque les puissances étaient évaluées avec trop d'erreurs lors du sous-cyclage du bain). En effet, ces puissances sont données par l'équation macroscopique de conservation de l'énergie pour chaque couche :

$$\phi^{\text{lat.}}_{\text{oxy.}} S^{\text{lat.}}_{\text{oxy.}} = m_{\text{oxy.}} C_{p_{\text{oxy.}}} \frac{\mathrm{d}T_{\text{oxy.}}}{\mathrm{d}t} + \dot{Q}_{\text{oxy.}} \tag{4.27}$$

$$\phi^{\text{lat.}}_{\text{metup.}} S^{\text{lat.}}_{\text{metup.}} = m_{\text{metup.}} C_{p_{\text{metup.}}} \frac{\mathrm{d}T_{\text{metup.}}}{\mathrm{d}t} + \dot{Q}_{\text{metup.}} \tag{4.28}$$

Le transitoire des puissances latérales et de thermiques des couches oxydes (oxy.) et métal léger (metup.) sont donnés par les figures 4.26a et 4.26b. Pour $t \leq 1700$ s, le transitoire de thermique est important du fait de l'établissement des phénomènes convectifs dominant largement le transitoire relativement lent de la puissance résiduelle et on a

$$\phi^{\text{lat.}}_{\text{oxy.}} S^{\text{lat.}}_{\text{oxy.}} \approx m_{\text{oxy.}} C_{p_{\text{oxy.}}} \frac{\mathrm{d}T_{\text{oxy.}}}{\mathrm{d}t} \gg \dot{Q}_{\text{oxy.}} \tag{4.29}$$

$$\phi^{\text{lat.}}_{\text{metup.}} S^{\text{lat.}}_{\text{metup.}} \approx m_{\text{metup.}} C_{p_{\text{metup.}}} \frac{\mathrm{d}T_{\text{metup.}}}{\mathrm{d}t} \gg \dot{Q}_{\text{metup.}} \tag{4.30}$$

et donc l'erreur dans l'évaluation des puissances résiduelles est masquée. Pour $t > 1700$ s, la convection est presque établie et un régime *quasi-stationnaire* de thermique est atteint. La puissance latérale déposée par le bain est alors

donnée par

$$\phi^{\text{lat.}}_{\text{oxy.}} S^{\text{lat.}}_{\text{oxy.}} \approx \dot{Q}_{\text{oxy.}} \gg m_{\text{oxy.}} C_{p_{\text{oxy.}}} \frac{\mathrm{d}T_{\text{oxy.}}}{\mathrm{d}t} \approx 0 \tag{4.31}$$

$$\phi^{\text{lat.}}_{\text{metup.}} S^{\text{lat.}}_{\text{metup.}} \approx \dot{Q}_{\text{metup.}} \gg m_{\text{metup.}} C_{p_{\text{metup.}}} \frac{\mathrm{d}T_{\text{metup.}}}{\mathrm{d}t} \approx 0 \tag{4.32}$$

et l'erreur sur l'évaluation de la puissance résiduelle n'est plus masquée.

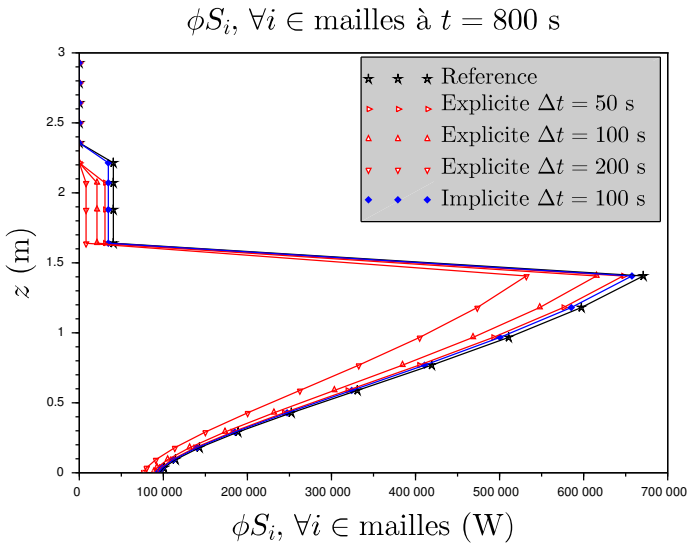
Comme on peut le voir sur la masse ablatée, cette erreur a un impact sur la résolution implicite. Néanmoins, la solution implicite reste tout de même plus proche de la solution de référence que la solution explicite. Pour l'instant $t = 800$ s de la première partie du transitoire et l'instant $t = 2600$ s de la seconde partie du transitoire, les figures 4.27a et 4.28a donnent les puissance ϕS_i déposées sur chaque maille i de la cuve et les figures 4.27b et 4.28b donnent les épaisseurs e_i de chaque maille i. Malgré l'erreur commise dans le modèle de bain dans la seconde partie du transitoire, la solution implicite coïncide avec la solution de référence et permet tout de même une meilleur évaluation de la puissance déposée sur la cuve par le bain et de l'épaisseur de la cuve pour les deux parties du transitoire.

Cependant, la résolution implicite reste moins précise pour le second cas test que pour le premier cas test. Les écarts en valeur absolu au cours du temps entre les solutions de référence et explicite et les solutions de référence et implicite (*bas*) pour $\Delta t = 100$ s sur l'épaisseur de la cuve (écart normalisé par rapport à l'épaisseur initiale $e^\star = 15$ cm de la cuve) sont donnés par les figures 4.29a et 4.29b. Alors que pour la résolution implicite du premier cas test (voir figure 4.23b), l'écart maximal était de 0.15 %, il est ici de près de 2 %. Cet écart, atteint dans la seconde partie du transitoire, est causé par l'erreur de l'ordre du macro pas de temps dans le modèle de bain. Pour la première partie du transitoire, l'écart reste faible (≤ 0.4 %).

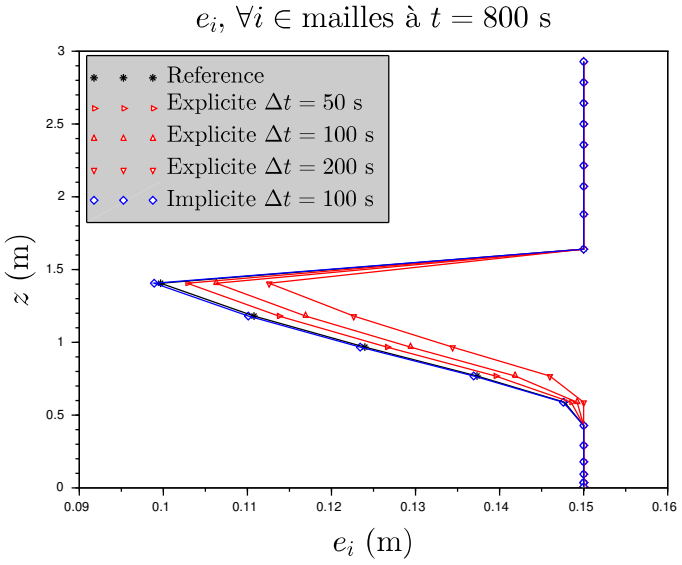
4.3.6.5 Conclusion

Dans le contexte de la stratégie IVR de rétention du corium en cuve, le couplage 4.10 entre le bain de corium et la cuve doit être résolu précisément. Ces résultats numériques montrent que la résolution implicite du couplage permet de retrouver la précision d'une résolution explicite avec un macro pas de temps faible. De plus, la précision obtenue est relativement peu sensible au macro pas de temps choisi.

Naturellement, plus le macro pas de temps est important, plus le nombre d'itérations du SCI pour chaque macro pas de temps sera conséquent. Différents schémas implicites ont été utilisés pour la résolution du couplage. Tous se basent sur un SCI de Gauss-Seidel par blocs avec relaxation (voir section 2.1). Pour un tel SCI avec paramètre de relaxation constant choisi correctement, le nombre d'itérations moyen est de 4 itérations et atteint au maximum

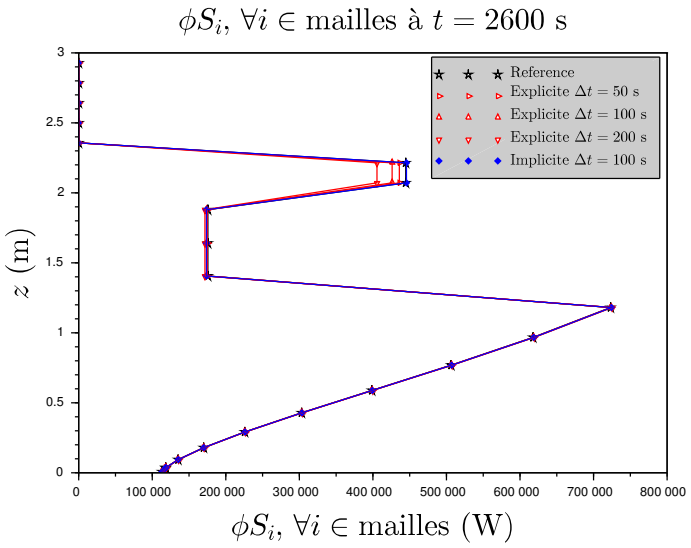


(a)

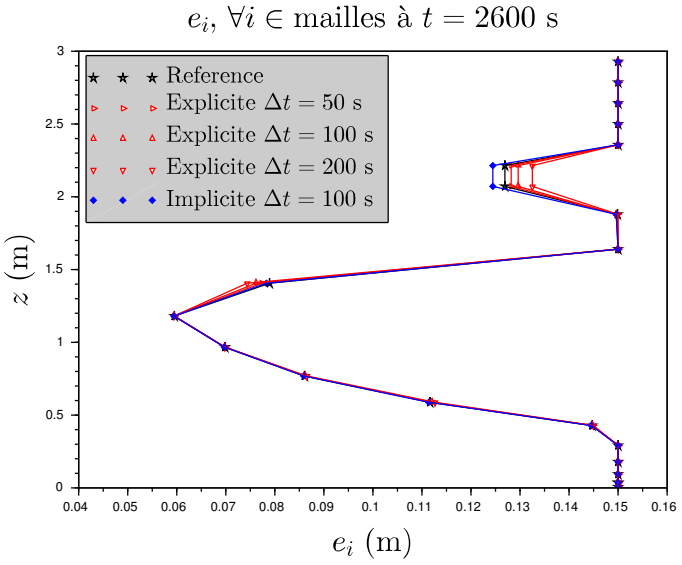


(b)

FIGURE 4.27 – **Second cas test** : à $t = 800$ s, puissances déposées sur chaque maille i par le bain de corium sur la cuve (*haut*) et épaisseurs de chaque maille i de la cuve (*bas*).

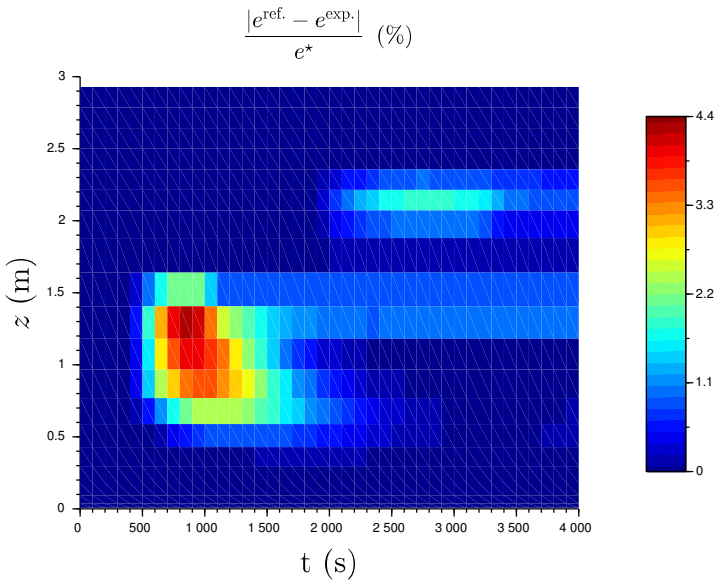


(a)

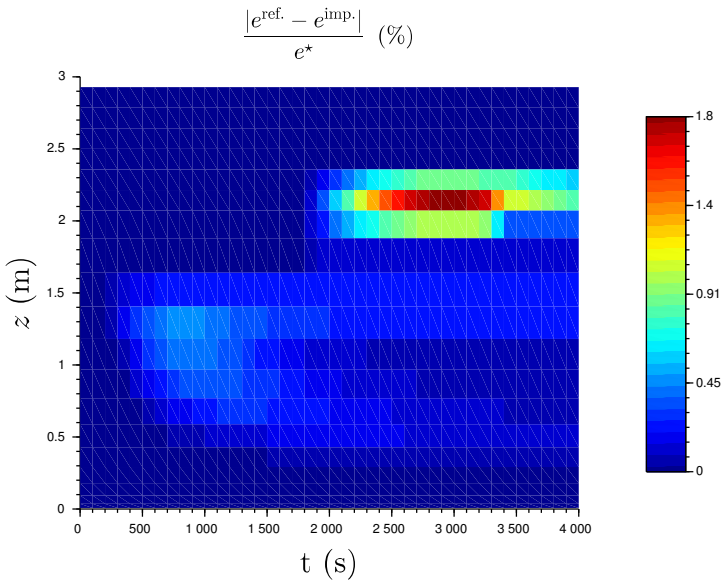


(b)

FIGURE 4.28 – **Second cas test** : à $t = 2600$ s, puissances déposées sur chaque maille i par le bain de corium sur la cuve (*haut*) et épaisseurs de chaque maille i de la cuve (*bas*).



(a)



(b)

FIGURE 4.29 – **Second cas test** : écart en valeur absolu au cours du temps entre les solutions de référence et explicite (*haut*) et les solutions de référence et implicite (*bas*) pour $\Delta t = 100$ s sur l'épaisseur de la cuve (écart normalisé par rapport à l'épaisseur initiale $e^\star = 15$ cm de la cuve).

15 itérations pour $\Delta t = 300$ s au début du transitoire. De plus, une stratégie de relaxation dynamique de type Aitken permet de rester inférieur à 5 itérations. En terme de coût de calcul et pour obtenir la même précision, on retrouve la conclusion apportée dans les premiers résultats section 4.1 :

— Pour un même Δt, la résolution implicite est plus coûteuse que la résolution explicite mais est *plus précise*. Néanmoins, la résolution implicite *peut être accélérée* et devient comparable à la résolution explicite en terme de temps de calculs.
— Permettant d'obtenir une *bonne précision* des calculs, la résolution implicite accélérée avec un macro pas de temps raisonnable est beaucoup plus *rapide* qu'une résolution explicite avec un macro pas de temps faible.

De plus, une autre conclusion que l'on peut tirer de ces tests est que la résolution implicite du couplage peut permettre d'identifier certains modèles générateurs d'erreur. Avant, la résolution explicite du couplage était toujours d'ordre 1 et les erreurs en interne des modèles pouvaient être cachées par l'erreur de couplage. Avec la résolution implicite, l'erreur de couplage est effacée et seules les erreurs commises en interne des modèles persistent. Si, après une résolution implicite du couplage, l'erreur est toujours de l'ordre du macro pas de temps, alors cela ne peut venir que de l'un des modèles du couplage. C'est ce que l'on a vu pour l'évaluation de la puissance du bain transmise à la cuve ou sur l'évaluation de la puissance résiduelle du bain : le modèle de bain de corium propageait une erreur de l'ordre du macro pas de temps dans le couplage qui ne pouvait être corrigée par le SCI.

On peut espérer étendre ces conclusions sur les autres modèles de la plateforme. Dans ce cas, une qualité logicielle accrue et des gains en précision des calculs des applications industrielles de la plate-forme PROCOR grâce à la nouvelle architecture de couplage seraient possibles.

4.3.7 Limites actuelles de l'application

Lors de tests sur l'application portée sous la nouvelle architecture, d'autres cas réacteurs et d'autres scénarios d'AG que celui utilisé précédemment [4] ont amenés les modèles de l'application dans des domaines physiques pour lesquels des discontinuités apparaissent. Malheureusement, ces discontinuités ne sont actuellement pas associées à des événements déclenchés par les modèles, généralement parce qu'elles n'avaient pas été clairement identifiées.

Un exemple d'une telle discontinuité est donnée dans la figure 4.30. Lors du transitoire de l'application [5], le volume du bain de corium en fond de cuve augmente par des ajouts de masse provenant de l'ablation de la cuve ou des lits de débris. Si l'on ne tient pas compte du transfert radiatif entre le bain de

4. Réacteur de Génération 3 (Gen3) avec un réflecteur lourd avec un scénario LOOP
5. Réacteur de Génération 3 (Gen3) avec un réflecteur lourd avec un scénario SBLOCA

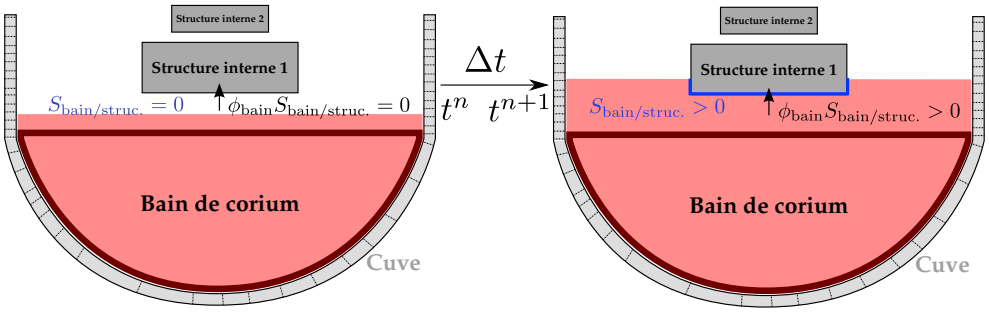
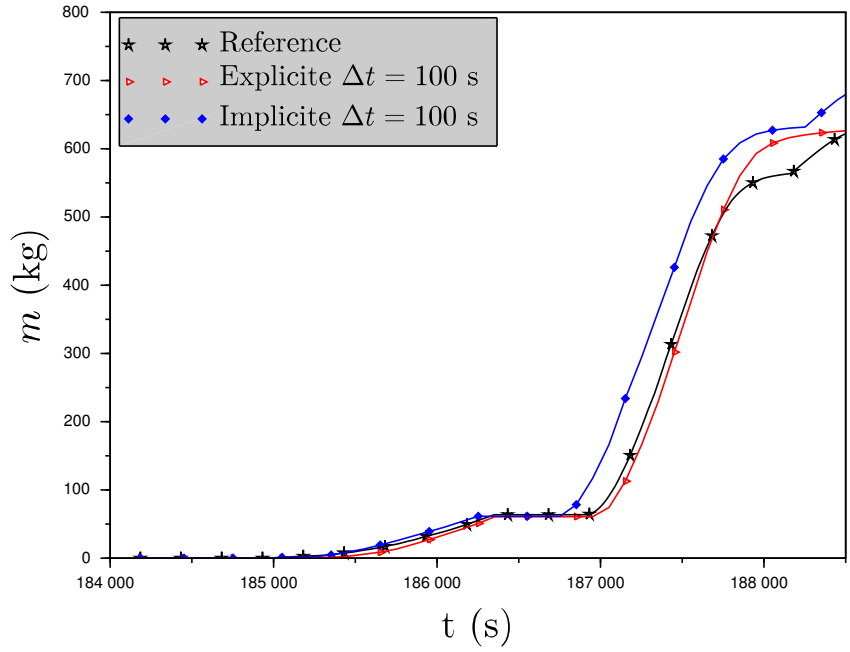


FIGURE 4.30 – Apparition d'une discontinuité sur la surface d'échange et donc dans la puissance déposée par le bain sur les structures internes lorsque celles-ci rentrent en contact avec le bain.

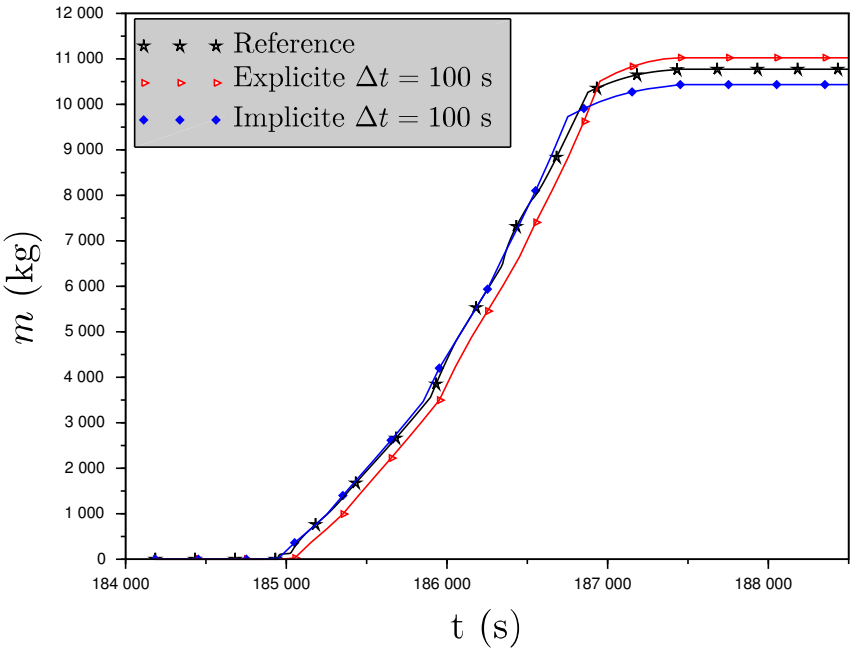
corium et les structures internes, la puissance déposée par le bain sur celles-ci est nulle (car la surface d'échange est nulle). Lorsqu'il y a contact, la puissance devient non nulle. Cependant, il est peu probable que le contact entre le bain et les structures se produise exactement sur un macro pas de temps. Généralement, le contact se produit à l'intérieur d'un macro pas de temps et la surface d'échange $S_{\text{bain/struc.}}$ entre le bain et les structures (en bleue dans la figure 4.30) passe de l'instant t^n à l'instant t^{n+1} d'une valeur nulle et à une valeur non nulle. De ce fait, bien que le flux de chaleur ϕ_{bain} du bain de corium est continu, la puissance $\phi_{\text{bain}} S_{\text{bain/struc.}}$ déposée sur les structures est discontinue.

Le contact entre le bain et les structures doit être un événement détecté et lancé par un modèle de l'application de telle sorte que le schéma de couplage puisse se synchroniser à l'instant de contact. Actuellement, ce n'est pas le cas. Par conséquent, le SCI et l'algorithme de synchronisation ne peuvent être conscients de la discontinuité associée au contact et ne peuvent se synchroniser à l'instant auquel elle apparaît. De ce fait, aucun traitement n'est fait au moment du passage par la discontinuité. À la discontinuité, la plupart des théorèmes mathématiques, par exemple ceux assurant la convergence des itérations de Newton, ne sont plus applicables et il est possible que les itérations de la résolution implicite ne convergent pas.

C'est ce qui est illustré dans les figures 4.31a et 4.31b donnant respectivement la masse ablatée de cuve et la masse des structures internes pour une solution de référence (*noire*), une résolution explicite (*rouge*) et une résolution implicite (*bleue*). La discontinuité apparaît entre les instants $t = 186652$ s et $t = 186752$ s de la macro boucle en temps $\Delta t = 100$ s. Avant la discontinuité, c'est à dire pour $t \leq 186652$ s, la solution implicite coïncide avec la solution de référence. Lors du macro pas de temps auquel la discontinuité apparaît, le SCI ne converge pas, oscille entre plusieurs valeurs et s'arrête lorsqu'il a atteint un nombre maximal d'itérations. De ce fait, le couplage n'est pas correctement



(a) Masse ablatée de la cuve



(b) Masse ablatée des structures internes

FIGURE 4.31 – Effets de la discontinuité de la puissance déposée par le bain sur les structures internes sur la masse ablatée de la cuve et des structures internes : non convergence de la résolution implicite à $t = 186652$ s puis création et propagation d'une erreur.

résolu et une erreur se créée et se propage jusqu'à la fin de la simulation.

Cet exemple montre l'importance de la *synchronisation* pour assurer le bon fonctionnement des SCI. De plus, il permet d'illustrer un résultat théorique (la convergence des itérations de Newton pour des fonctions continues) dans un cas réel d'utilisation d'une application industrielle AG.

Enfin, de cet exemple et des résultats précédents, plusieurs pistes d'améliorations de l'application industrielle `PROCOR TransientInLowerHead` sont ouvertes.

Il faudrait diminuer les erreurs commises en internes des modèles couplés constituant l'application en résolvant plus précisément le partitionnement dans le modèle du bain de corium.

De plus, il faudrait identifier les discontinuités présentes dans les modèles couplés constituant l'application, par exemple celle sur la puissance imposée par le bain de corium sur les structures internes, pour pouvoir se synchroniser dessus.

Néanmoins, se synchroniser sur toutes les discontinuités peut devenir très coûteux et pénaliser très fortement les performances de calculs du schéma de couplage. Il faudrait déjà traiter et éliminer une partie des discontinuités dans les modèles. Certaines discontinuités ne peuvent être éviter. Cependant, certaines discontinuités non-physiques présentes dans les modèles peuvent être traitées et, pour d'autres, il est possible de *régulariser* le modèle en améliorant la physique au sein du modèle. En reprenant l'exemple précédent, on pourrait modéliser un *transfert radiatif* entre le bain de corium et les structures internes : au contact entre le bain et les structures, la puissance ne passe pas d'une valeur nulle à une valeur importante et on réduit ainsi la discontinuité. De plus, on pourrait imaginer modéliser une interface diffuse lorsque le bain et les structures sont suffisamment proches (ce qui garde un sens physique car en réalité la surface supérieure du bain n'est pas plane) à l'intérieur de laquelle la puissance augmente *continûment*.

Conclusions et perspectives

Conclusions

Ce travail de thèse a porté sur la résolution numérique *partitionnée* et *synchronisée* des *systèmes complexes* relatifs à la simulation d'accidents graves (AG) dans les réacteurs à eau légère au sein de la plate-forme logicielle PROCOR.

Les systèmes complexes mis en jeu dans la simulation des accidents graves sont découpés en blocs couplés sur une macro boucle en temps, c'est l'approche partitionnée de résolution des problèmes couplés sous-jacents aux systèmes complexes. Néanmoins, les efforts initiaux de développement dans la plate-forme PROCOR ne se sont pas portés sur l'aspect logiciel et numérique du partitionnement. Initialement, dans la plate-forme, la résolution du partitionnement se faisait par simple chaînage des modèles. Cette résolution est connue pour être *peu précise*, *parfois instable* et n'offrant des possibilités de synchronisation que *très limitées*. Des calculs sur des applications industrielles de la plate-forme ont montré que, pour la gamme de macro pas de temps utilisés, le schéma de couplage utilisé était encore loin de son domaine de convergence et, par conséquent, une erreur, de l'ordre du macro pas de temps, se créait à chaque macro pas de temps et se propageait tout au long de la simulation. De plus, les possibilités de synchronisation offertes par la résolution par chaînage des modèles sont très limitées et peu précises. Une solution permettant une résolution précise du partitionnement et de la synchronisation est de diminuer le macro pas de temps de couplage. Néanmoins, ceci est très souvent impossible. Pour des raisons de performance, les calculs doivent rester rapides, par exemple dans le cadre d'études statistiques pour lesquelles le nombre de calculs peut devenir très important. De plus, certains modèles physiques de la plate-forme imposent l'utilisation d'un macro pas de temps important (modèle stationnaire par exemple).

Dans la section 2.1 du chapitre 2, un formalisme permettant de représenter les systèmes complexes apparaissant lors de la simulation des AG a été proposé. Il a permis d'introduire les algorithmes partitionnés permettant la résolution des problèmes couplés définis par ces systèmes complexes. Deux classes d'algorithmes de résolution du partitionnement ont été présentées :

les schémas de couplage explicite et les schémas de couplage implicite. L'algorithme de résolution explicite CSS, correspondant à la résolution chaînée des modèles utilisée dans la plate-forme PROCOR, a été détaillé. En particulier, on a vu que l'erreur commise à chaque macro pas de temps par l'explicitation du couplage est proportionnelle, entre autres, à la longueur du macro pas de temps et à la force du couplage physique entre les modèles. De ce fait, pour des macro pas de temps élevés imposés par le contexte industriel et/ou par le contexte physique et pour des couplages forts entre modèles, l'erreur peut devenir importante. On a par la suite présenté les algorithmes de partitionnement implicites. Dans cette thèse, on s'est concentré sur les algorithmes de type Gauss-Seidel par blocs avec relaxation offrant un bon compromis entre la simplicité de mise en œuvre, la précision et le coût des calculs. Les algorithmes de partitionnement implicites permettent d'obtenir, au fur et à mesure des itérations sur un macro pas de temps, un couplage fort entre modèles contrôlé par la tolérance numérique du schéma.

Dans la section 2.2 du chapitre 2, un algorithme de synchronisation des modèles a été présenté. Intégré aux algorithmes de partitionnement implicites, il permet une synchronisation précise sur des événements déclenchés par les modèles à l'intérieur du macro pas de temps (événements souvent associés à des discontinuités des modèles pouvant nuire à la précision des calculs). Ainsi proposé, l'algorithme de partitionnement et de synchronisation permet, à la fin de chaque macro pas de temps, d'obtenir un couplage fort entre modèles et une synchronisation sur les événements de modèles. Les discontinuités des modèles sont ainsi localisées à chaque macro pas de temps permettant aux solveurs et aux fonctions mathématiques sous-jacentes d'être régulières sur chaque macro pas de temps.

Une des contributions les plus importantes de cette thèse est l'intégration dans la plate-forme industrielle PROCOR d'une architecture logicielle de couplage *modulaire* et *évolutive*. Elle est présentée dans le chapitre 3. Elle permet d'implémenter progressivement les anciens modèles de la plate-forme sous ce nouveau paradigme tout en réutilisant l'ancien schéma de couplage explicite de chaînage des modèles. On peut ainsi retrouver les précédents résultats des modèles assurant le respect des contraintes de non régression imposées par la contexte industriel de la plate-forme. En plus, les modèles peuvent utiliser toutes les fonctionnalités offertes par la plate-forme. Par exemple, utiliser les algorithmes de partitionnement implicites et/ou l'algorithme de synchronisation. De plus, la plate-forme répond maintenant parfaitement aux contraintes de modélisation associées au contexte AG : les modèles peuvent être facilement branchés et débranchés d'un problème couplé pour être substitués par d'autres modèles.

Enfin, des résultats numériques donnés dans le chapitre 4 fournis par des calculs de la plate-forme PROCOR ont permis de mettre en valeur le poten-

tiel de la nouvelle architecture de couplage. Dans un premier temps, on s'est concentré sur des problèmes couplés clés de la plate-forme puis, dans un second temps, on a donné des résultats issus d'une application industrielle de la plate-forme PROCOR. Au travers de ces résultats, on a pu voir que la nouvelle architecture de couplage a eu un fort impact positif *numérique* et *logiciel* sur la plate-forme PROCOR.

Impact numérique de l'architecture de couplage. La résolution du partitionnement implicite et synchronisée permet une résolution précise du couplage en modèles tout en gardant un macro pas de temps de couplage important. Ainsi, les contraintes imposées par le contexte industriel et physique sur le macro pas de temps peuvent être respectées. Néanmoins, on a vu que, même si le couplage est résolu précisément, l'erreur en interne des modèles ne peut être corrigée par la résolution implicite du couplage. On a vu de plus que cette erreur peut être importante auquel cas la résolution implicite n'a pas d'intérêt par rapport à la résolution explicite. Un travail numérique et de modélisation physique doit être fait en interne du modèle incriminé.

Impact logiciel de l'architecture de couplage. L'utilisation de l'architecture de couplage pour les anciens modèles de la plate-forme a un impact positif sur la *qualité* du logiciel PROCOR. Lors de la ré-implémentation des modèles sous le nouveau paradigme de l'architecture de couplage, les entrées, les sorties et différents couplages entre modèles doivent être définis. Ceci a permis de révéler de nombreuses erreurs commises parce que celles-ci étaient avant traitées au cas par cas et de manière *ad-hoc*. De plus, on a vu que l'utilisation de schémas implicites permet la suppression de portions de codes uniquement destinés à la correction de situations non physiques amenées par la résolution explicite du couplage. Ainsi, une véritable séparation de métier est faite entre le modélisateur physicien s'occupant des modèles physiques et le numéricien s'occupant de la résolution du couplage.

Maintenant, la plate-forme PROCOR peut être vue comme un environnement de couplage répondant parfaitement aux contraintes du contexte très spécifique des AG. D'une part en termes de modularité, un modèle peut par exemple être échangé facilement par un autre modèle plus évolué ou utilisant une modélisation différente au sein d'une application industrielle. D'autre part en termes de choix du schéma de couplage, on peut choisir le schéma de couplage adapté au cas d'utilisation de la plate-forme, par exemple pour une étude précise et fine ou pour une étude industrielle ne nécessitant que des résultats enveloppes, ou adapté au scénario d'accident, ne nécessitant parfois qu'un couplage faible entre modèles et parfois un couplage fort entre modèles. Les objectifs de la thèse ont tous été remplis. L'architecture de couplage ouvre maintenant d'autres pistes possibles de travail.

Perspectives

L'optimisation du parcours du graphe de couplage par les SCI est une des pistes principales de travail à la suite de la thèse. La force du couplage entre les différents modèles d'un problème couplé n'est pas toujours la même. Certains couplages sont plus forts que d'autres. Par exemple, les résultats numériques présentés chapitre 4 ont permis d'identifier que le couplage entre le bain de corium et la cuve est plus fort que les autres et il doit être absolument résolu de manière implicite. À l'inverse, d'autres couplages, par exemple entre le modèle d'eau et les autres modèles, sont moins forts et peuvent n'être résolus qu'explicitement. Il serait possible de détecter automatiquement la force du couplage, par exemple en approchant la jacobienne de l'équation de point fixe (2.10) de l'interface (associée à la force du couplage) par les résidus (2.11), et d'adapter le schéma de couplage en conséquence. De l'*intelligence* serait ajoutée au schéma de couplage. Le schéma s'adapterait automatiquement et dynamiquement aux couplages du problème qui peuvent évoluer *au cours* de la simulation. On peut aussi imaginer mettre des poids sur les arrêtes du graphe en fonction de la force du couplage correspondant à l'arrête. Le parcours du graphe pourrait ensuite être optimisé en fonction des poids sur les arrêtes. En détectant les composantes connexes du graphe, on peut aussi paralléliser simplement les calculs.

Une autre piste constituant une perspective possible à la suite de cette thèse est l'utilisation du *parallélisme* pour accélérer les résolutions implicites du couplage. On pense notamment aux algorithmes "parareal" développés dans [73] et approfondis dans [40, 39]. Par l'utilisation du graphe de couplage des modèles, il serait possible de détecter des composantes non connectées qui peuvent être résolues en parallèle. Ces algorithmes ont notamment été utilisés pour accélérer la résolution de problèmes couplés avec des méthodes types "Waveform Relaxation" dans [71] (les méthodes "Waveform Relaxation" sont des méthodes itératives de résolution de problèmes couplés consistant à envoyer, à chaque macro pas de temps, des données temporelles entre les modèles couplés. Elles sont particulièrement adaptées lorsque les modèles sont fortement sous-cyclés. De nombreux résultats théoriques sont disponibles, notamment sur la convergence super linéaire des itérations. Plus de détails peuvent être trouvés dans [55, 7, 18, 52].)

Il serait aussi possible d'améliorer la résolution des modèles avec équations différentielles algébriques (EDA) de la plate-forme PROCOR, présentés 3.2. Les EDA, formées des équations différentielles ordinaires (EDO) de conservation de la masse et de l'énergie (ou de l'enthalpie) et des équations algébriques (EA) de fermetures (voir l'annexe A pour des exemples de lois de fermeture), sont actuellement résolus par des solveurs EDO. Or, ces solveurs ne sont pas adaptés (voir [82] par exemple). Il faudrait faire un inventaire des différentes EDA présentes puis les classer suivant leur indice. Celui-ci per-

met ensuite de choisir le solveur adapté. Ensuite, il ne resterait plus qu'à implémenter le ou les solveurs dans la plate-forme. Une comparaison entre la résolution actuelle et la résolution avec les nouveaux solveurs serait ensuite possible.

La plate-forme PROCOR étant organisée en package Java relativement indépendant, il serait envisageable de détacher le noyau de PROCOR contenu dans le package `kernel` (présenté dans la section 3.2) et notamment les classes relatives à l'architecture de couplage pour en faire un module externe. Ainsi, l'architecture pourrait être utiliser hors de PROCOR dans d'autres plates-formes. Par exemple pour la plate-forme SEASON. En cours de développement au sein du même laboratoire (DEN DTN/SMTA/LMAG) que celui de la thèse, cette plate-forme doit est utilisée dans le cadre des AG dans les réacteurs à neutrons rapides. Elle permet le couplage de différents codes (par exemple le code de thermohydraulique SIMMER [63]). La résolution du couplage entre les codes et modèles de la plate-forme SEASON pourrait être faite par l'architecture de couplage de PROCOR. Elle viendrait s'intégrer dans SEASON comme une *bibliothèque externe*.

Dans les études probabilistes de sûreté, de *fortes hypothèses* sont faites sur la réponse des équipements et des opérateurs et sur le temps des événements survenant lors de l'accident. L'arbre des événements associés au scénario de l'accident est créé à partir de ces hypothèses. Chaque embranchement correspond à un temps fixé ou un événement extérieur (par exemple l'intervention d'un opérateur, l'ouverture d'une vanne). À chaque embranchement de l'arbre, des calculs sont lancés en partant du même état sans prise en compte de ce qu'il s'était passé avant. Cette approche *statique* d'études probabilistes de sûreté ne tient pas en compte de la dynamique du système, de l'interaction des différents composants du système complexe (composante humaine ou physique) et en particulier du temps d'apparition des événements. Dans une approche *dynamique de génération de l'arbre des événements* (ou "Dynamic Event Trees" dans [89, 49, 2, 59]) de l'accident, les temps des événements sont donnés par la dynamique du système permettant de générer *dynamiquement* l'arbre des événements à partir d'un seul calcul (partant de l'événement initiateur de l'accident). À chacun de ces événement, sur lequel le système doit se *synchroniser* précisément, une nouvelle branche est créée à partir d'un état du système cohérent avec la dynamique du système et avec les événements antérieurs. Pour chaque nouvelle branche, un nouveau calcul est lancé (en séquentiel ou en parallèle) à partir d'états cohérents. Cependant, ces méthodes dynamiques sont beaucoup plus *intrusives* que les méthodes statiques de PSA. Elle nécessite, à chaque événement, la *sauvegarde* de l'état courant pour pouvoir lancer séquentiellement ou parallèlement de nouveaux calculs à partir de cet état. De plus, la *synchronisation* sur les événements est un point crucial de telles méthodes. Grâce à l'architecture intégrée dans la plate-forme PROCOR pour la thèse, ces fonctionnalités sont possibles et des études probabilistes de

sûreté *dynamiques* sont maintenant *envisageables*.

Enfin, il serait intéressant d'étudier l'impact du schéma de couplage et de synchronisation sur la précision en *arithmétique flottante* des calculs de la plate-forme PROCOR. La résolution implicite et synchronisée du couplage améliore la précision numérique des calculs mais, a priori, rien n'est certain sur le gain qu'elle apporte en arithmétique flottante (les itérations des SCI pourraient par exemple amplifiées les erreurs flottantes contrairement aux SCE faisant moins d'appels aux solveurs et propageant peut être moins d'erreurs). De plus, les modèles à seuil et les modèles avec événements, largement utilisés dans PROCOR, sont connus pour être sensibles à la précision en arithmétique flottante, rendant l'étude d'autant plus intéressante pour la plate-forme. Des outils tels que `Verificarlo` [23] ou `Verrou` [25] (déjà en test dans PROCOR) pourraient être utilisés.

Annexe A

Construction des modèles 0D

On décrit les étapes d'intégration des équations locales de conservation sur chaque domaine Ω_i de la figure 2 de la section 4.1 permettant d'obtenir les modèles OD ou "lumped parameter" utilisés section 4.1. On considère un domaine liquide Ω_l et un domaine solide Ω_s. Ils sont séparés par une interface mobile Γ_{ls}. Dans cette annexe, une condition de Stefan est imposée à l'interface associée à un front de fusion solidification entre les deux domaines.

A.1 Équations locales de conservation

Pour le domaine liquide $\Omega_l(t)$, les équations de conservation de la masse, de l'énergie et de la quantité de mouvement sont écrites en fonction de la vitesse $\vec{v}_l$ et de la température T_l. Elles sont données sous les hypothèses de Boussinesq par

$$\vec{\nabla} \cdot \vec{v}_l(\vec{r}, t) = 0 \tag{A.1}$$

$$\rho_l^{\text{ref.}} C_{p_l} \left(\frac{\partial T_l}{\partial t}(\vec{r}, t) + \vec{v}_l \cdot \vec{\nabla} T_l(\vec{r}, t) \right) = \lambda_l \Delta T_l(\vec{r}, t) \tag{A.2}$$

$$\begin{aligned} \frac{\partial \vec{v}_l}{\partial t}(\vec{r}, t) + \vec{v}_l \cdot \vec{\nabla} \vec{v}_l(\vec{r}, t) = \frac{-1}{\rho_l^{\text{ref.}}} \vec{\nabla} p(\vec{r}, t) + \nu_l \Delta \vec{v}_l(\vec{r}, t) \\ - \beta_l \left(T_l(\vec{r}, t) - T_l^{\text{ref.}} \right) \vec{g} \end{aligned} \tag{A.3}$$

pour $\vec{r} \in \Omega_l(t)$ avec des conditions initiales fixées (en particulier, la température $T_l(\vec{r}, 0) = T_l^0(\vec{r})$). Les conditions aux bords du domaine données par

$$\vec{v}_l(\vec{r}, t) \cdot \vec{n} = 0, \quad \forall \vec{r} \in \partial\Omega_l(t) \setminus \Gamma_{ls}(t) \tag{A.4}$$

$$\text{conditions en } T_l(\vec{r}, t) \text{ et/ou } -\lambda_l \vec{\nabla} T_l(\vec{r}, t) \cdot \vec{n}, \quad \forall \vec{r} \in \partial\Omega_l(t) \setminus \Gamma_{ls}(t) \tag{A.5}$$

$$T_l(\vec{r}, t) = T^{\text{fus.}}, \quad \forall \vec{r} \in \Gamma_{ls}(t). \tag{A.6}$$

Les propriétés physiques du domaine liquide sont : la masse volumique $\rho_l^{\text{ref.}}$ à une température de référence $T_l^{\text{ref.}}$, la viscosité dynamique ν_l, le coefficient de dilatation thermique β_l, la capacité calorifique C_{p_l} et la conductivité thermique λ_l.

Dans le domaine solide $\Omega_s(t)$, l'évolution de la température T_s est gouvernée par l'équation de conduction de la chaleur donnée par

$$\rho_s C_{p_s} \frac{\partial T_s}{\partial t}(\vec{r}, t) = \lambda_s \Delta T_s(\vec{r}, t), \quad \forall \vec{r} \in \Omega_s(t). \tag{A.7}$$

avec la condition initiale $T_s(\vec{r}, 0) = T_s^0(\vec{r})$. Les conditions aux bords du domaine sont données par

$$\text{conditions en } T_s(\vec{r}, t) \text{ et/ou } -\lambda_s \vec{\nabla} T_s(\vec{r}, t), \quad \forall \vec{r} \in \partial\Omega_s \setminus \Gamma_{ls}(t) \tag{A.8}$$

$$T_s(\vec{r}, t) = T^{\text{fus.}}, \quad \forall \vec{r} \in \Gamma_{ls}(t). \tag{A.9}$$

Les propriétés physiques du domaine solide sont : la masse volumique ρ_s, la capacité calorifique C_{p_s} et la conductivité thermique λ_s.

Enfin, la conservation de la masse à l'interface est donnée par

$$\rho_s \vec{v}^{\text{ls}}(\vec{r}, t) \cdot \vec{n}^{\text{ls}} = \rho_l \left(\vec{v}^{\text{ls}}(\vec{r}, t) - \vec{v}_l(\vec{r}, t) \right) \cdot \vec{n}^{\text{ls}}, \quad \forall \vec{r} \in \Gamma_{ls}(t) \tag{A.10}$$

et la vitesse de l'interface liquide-solide Γ_{ls} est décrite par un front de fusion plan donné par

$$\vec{v}^{\text{ls}}(\vec{r}, t) = \frac{1}{\rho_s \Delta \mathcal{H}^{\text{fus.}}} \left(-\lambda_l \vec{\nabla} T_l(\vec{r}, t) \cdot \vec{n}^{\text{ls}} + \lambda_s \vec{\nabla} T_s(\vec{r}, t) \cdot \vec{n}^{\text{ls}} \right) \vec{n}^{\text{ls}}, \quad \forall \vec{r} \in \Gamma_{ls}(t) \tag{A.11}$$

avec $\vec{n}^{\text{ls}}$ orienté du liquide vers le solide. Les propriétés physiques de fusion sont : la température de fusion $T^{\text{fus.}}$ et la chaleur latente de fusion $\Delta \mathcal{H}^{\text{fus.}}$.

A.2 La formulation 0D ou "lumped parameter" (LP)

À partir de ces équations, en écartant l'équation de quantité de mouvement, le problème dans les deux phases peut être intégré permettant d'obtenir des équations macroscopiques de conservation de la masse et de l'énergie. Cette approche mène à la formulation 0D ou "lumped parameter" (LP).

Pour ce faire, la frontière $\partial\Omega_l(t)$ est partitionnée sous la forme

$$\partial\Omega_l(t) = \left(\bigcup_i \partial\Omega_{l,i} \right) \bigcup \Gamma_{ls}(t)$$

et les quantités intégrales suivantes sont définies :

$$m_l(t) \triangleq \rho_l V_l(t) \tag{A.12}$$

$$\bar{T}_l(t) \triangleq \frac{1}{V_l(t)} \int_{\Omega_l(t)} T_l(\vec{r},t) dV \tag{A.13}$$

$$\dot{m}^{\text{ls}}(t) \triangleq \rho_s \int_{\Gamma_{ls}} \vec{v}^{\text{ls}}(\vec{r},t) \cdot \vec{n}^{\text{ls}} dS \tag{A.14}$$

$$\bar{\phi}_l^i(t) \triangleq -\frac{1}{S_l^i} \int_{\partial\Omega_{l,i}} \lambda_l \vec{\nabla} T_l(\vec{r},t) \cdot \vec{n} dS \tag{A.15}$$

$$\bar{\phi}_l^{ls}(t) \triangleq -\frac{1}{S_{ls}(t)} \int_{\Gamma_{ls}(t)} \lambda_l \vec{\nabla} T_l(\vec{r},t) \cdot \vec{n}^{\text{ls}} dS \tag{A.16}$$

avec S_{ls} (resp. S_l^i) la surface de Γ_{ls} (resp. $\partial\Omega_{l,i}$).

Ainsi, en utilisant le théorème de transport de Reynolds et le théorème de Gauss, la formulation intégrale des équations (A.1) et (A.2) est donnée par

$$\frac{dm_l(t)}{dt} = \dot{m}^{\text{ls}}(t) \tag{A.17}$$

$$C_{p_l}\left(m_l(t)\frac{d\bar{T}_l}{dt}(t) + \dot{m}^{\text{ls}}(t)\left(\bar{T}_l(t) - T^{\text{fus.}}\right)\right) = -\sum_i \bar{\phi}_l^i(t) S_l^i - \bar{\phi}_l^{ls}(t) S_{ls}(t) \tag{A.18}$$

Le même processus d'intégration peut être mené pour le domaine solide pour obtenir la formulation intégrale suivante :

$$\frac{dm_s(t)}{dt} = -\dot{m}^{\text{ls}}(t) \tag{A.19}$$

$$C_{p_s}\left(m_s(t)\frac{d\bar{T}_s}{dt}(t) + \dot{m}^{\text{ls}}(t)\left(T^{\text{fus.}} - \bar{T}_s(t)\right)\right) = -\sum_i \bar{\phi}_s^i(t) S_s^i + \bar{\phi}_s^{ls}(t) S_{ls}(t) \tag{A.20}$$

avec les mêmes notations que pour le domaine liquide, le flux de chaleur par conduction est donné par

$$\bar{\phi}_s^{ls}(t) \triangleq -\frac{1}{S_{ls}(t)} \int_{\Gamma_{ls}(t)} \lambda_s \vec{\nabla} T_s(\vec{r},t) \cdot \vec{n}^{\text{ls}} dS \tag{A.21}$$

Enfin, l'équation d'interface (A.11) est donnée par

$$\dot{m}^{\text{ls}}(t) = \frac{S_{ls}(t)}{\Delta\mathcal{H}^{\text{fus.}}}\left(\bar{\phi}_l^{ls}(t) - \bar{\phi}_s^{ls}(t)\right) \tag{A.22}$$

Naturellement, l'utilisation de telles équations 0D nécessite des lois de fermeture ou des conditions de Neumann aux frontières associées aux flux de chaleur moyennés $\bar{\phi}_l^i(t)$ et $\bar{\phi}_s^i(t)$.

Par exemple, on utilise pour le domaine liquide des lois de fermeture sous la forme de corrélations de Nusselt-Rayleigh (voir [12] ou [95] par exemple) permettant de modéliser les phénomènes de convection naturelle. Ces fermetures sont détaillées par la suite.

Pour le domaine solide, différents modèles existent pour calculer les lois de fermeture. Ci-après, on présente la construction d'un tel modèle. Plus de détails sur celui-ci peuvent être trouvés dans [70].

A.2.1 Fermetures du domaine liquide

On décrit ici une manière permettant d'obtenir des lois de fermeture pour le domaine liquide. Naturellement, d'autres voies sont possibles.

Les flux de chaleur pour le domaine liquide permettent de modéliser un problème de convection naturelle. Les flux de chaleur sont calculés à partir des coefficients de transfert thermique h. Ceux-ci sont liés au nombre de Nusselt Nu par :

$$h_l^{ls}(t) = \frac{\lambda_l Nu_l^{ls}(t)}{e_l(t)} \tag{A.23}$$

$$h_l^{i}(t) = \frac{\lambda_l Nu_l^{i}(t)}{e_l(t)} \tag{A.24}$$

avec λ_l la conductivité thermique du liquide et e_l la longueur caractéristique du liquide. Ensuite les flux de chaleurs $\bar{\phi}_l^i$ et $\bar{\phi}_l^{ls}$ dans le domaine liquide sont donnés par les équations

$$\bar{\phi}_l^{ls}(t) = h_l^{ls}(t)\left(\bar{T}_l(t) - T^{\text{fus.}}\right) = \frac{\lambda_l Nu_l^{ls}(t)}{e_l(t)}\left(\bar{T}_l(t) - T^{fus.}\right) \tag{A.25}$$

$$\bar{\phi}_l^{i}(t) = h_l^{i}(t)\left(\bar{T}_l(t) - T_l^i\right) = \frac{\lambda_l Nu_l^{i}(t)}{e_l(t)}\left(\bar{T}_l(t) - T^i\right). \tag{A.26}$$

avec :

- g l'accélération de la pesanteur en m.s^{-2}.
- β_l le coefficient de transfert de chaleur convectif en K^{-1}.
- $\dot{q}_l^{vol}$ la puissance volumique en W.m^{-3}.
- $\dot{q}_l$ la puissance massique en W.kg^{-1}.
- α_l la diffusivité thermique en m^2.s^{-1}.
- ν_l la viscosité dynamique en Pa.s.
- λ_l la conductivité thermique en W.m^{-1}.K^{-1}.

Le nombre de Nusselt représente le rapport entre les phénomènes convectifs et les phénomènes conductifs dans le liquide. On peut montrer qu'il est associé à la quantité de mouvement dans le liquide. Utiliser ce nombre pour calculer les flux de chaleur (qui sont ensuite utilisés dans l'équation de conservation d'énergie (A.18)) permet de prendre en compte la *quantité de mouvement* dans le liquide au travers de l'équation d'énergie (la quantité de mouvement n'est pas prise en compte explicitement dans les équations intégrales (A.17) et (A.18) du liquide).

Pour la convection naturelle, la nombre de Nusselt Nu peut par exemple être calculé à partir de corrélations dépendante du nombre de Rayleigh Ra et du nombre de Prandtl $Pr = \nu_l/\alpha_l$. Ces corrélations prennent généralement la forme de fonctions généralement non-linéaires. Deux types de corrélation existent (voir [12, 95]) suivant la nature du liquide, changeant la manière de calculer le nombre de Rayleigh :

Pour les fluides avec puissance interne Un nombre de Rayleigh interne Ra_i est utilisé. Ce nombre, peu connu hors de la communauté AG, fait notamment intervenir la puissance interne du liquide (ou puissance résiduelle dans le cas d'un liquide radioactif). Celui ci est donné par :

$$Ra_i(t) = \frac{g\,\beta_l\,\dot{q}_l^{vol}(t)\,e_l(t)^5}{\alpha_l\,\nu_l\,\lambda_l} = \frac{g\,\beta_l\,\rho_l\,\dot{q}_l(t)\,e_l(t)^5}{\alpha_l\,\nu_l\,\lambda_l} \tag{A.27}$$

Pour les fluides sans puissance interne Un nombre de Rayleigh externe Ra_e est utilisé. Le nombre de Rayleigh externe fait quant à lui intervenir la différence de température entre la température du liquide et la température l'interface du domaine considéré :

$$Ra_e(t) = \frac{g\,\beta_l\,e_l(t)^3\,(T_l(t) - T_l^i)}{\alpha_l\,\nu_l} \tag{A.28}$$

De manière générale, le nombre de Nusselt vaut toujours

$$Nu(t) = f(Ra_{\{i,e\}}(t), Pr)$$

avec f une fonction non linéaire. On utilise généralement une fonction de la forme $f(Ra, Pr) = a\,Ra^b$ avec $a, b \in \mathbb{R}$ issus de corrélations.

A.2.2 Fermetures du domaine solide

On décrit ici un modèle permettant d'obtenir des lois de fermeture pour le domaine solide. Naturellement, d'autres modèles sont possibles.

On fait l'hypothèse d'un front de fusion 1D comme décrit dans la figure A.1.

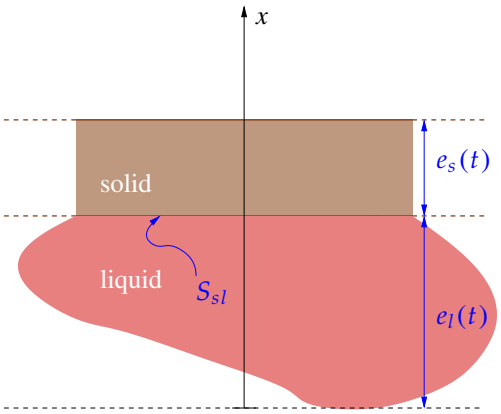


FIGURE A.1 – Approximation 1D du front de fusion et notations associées

À part à l'interface avec le solide, aucune hypothèse n'est faite sur la géométrie du liquide. L'épaisseur $e_s(t)$ du domaine solide est donnée par

$$e_s = \frac{m_s}{S_{ls}\rho_s}.$$

La frontière extérieure du domaine liquide est choisie comme étant l'origine de l'axe des x, la coordonnée de l'interface liquide-solide est donnée par $e_l(t)$ et la frontière extérieure du solide est à

$$x^s(t) = e_l(t) + e_s(t).$$

Dans ce cas, les équations (A.7), (A.8) et (A.9) du domaine solide se simplifient en

$$\rho_s C_{p_s} \frac{\partial T_s}{\partial t}(x,t) = \lambda_s \frac{\partial^2 T_s}{\partial x^2}(x,t), \quad \forall x \in]e_l(t), x^s(t)[\tag{A.29}$$

$$\text{conditions en } T_s(x,t) \text{ et/ou } -\lambda_s \frac{\partial T_s}{\partial x}(x,t), \quad x = x^s(t) \tag{A.30}$$

$$T_s(x,t) = T^{\text{fus.}}, \quad x = e_l(t) \tag{A.31}$$

De la même manière, l'équation 0D (A.20) de conservation dans le domaine solide est

$$C_{p_s}\left(m_s \frac{d\bar{T}_s}{dt} + \dot{m}^{\text{ls}}\left(T^{\text{fus.}} - \bar{T}_s\right)\right) = \left(-\bar{\phi}_s^{bc} + \bar{\phi}_s^{ls}\right) S \tag{A.32}$$

avec $S = S_{ls} = S_s^{bc}$ et les flux de chaleur moyens sont :

$$\bar{\phi}_s^{ls}(t) = -\lambda_s \frac{\partial T_s}{\partial x}(e_l(t), t) \tag{A.33}$$

$$\bar{\phi}_s^{bc}(t) = -\lambda_s \frac{\partial T_s}{\partial x}(x^s(t), t) \tag{A.34}$$

Ensuite, l'équation (A.29) peut être *directement* résolue (par exemple par différences finies) et les flux de chaleur $\bar{\phi}_s^{ls}(t)$ et $\bar{\phi}_s^{bc}(t)$ peuvent être calculés par les équations (A.33) et (A.34).

Permettant d'outrepasser la résolution relativement coûteuse de (A.29), un autre choix consiste à faire des hypothèses de modélisation sur la température dans le solide pour calculer les flux de chaleur $\bar{\phi}_s^{ls}(t)$ et $\bar{\phi}_s^{bc}(t)$. Par exemple dans [70], on fait l'hypothèse d'un profil quadratique pour T_s. Cela permet le calcul des flux de chaleur par les équations (A.33) et (A.34) par

$$\bar{\phi}_s^{ls}(t) = \lambda_s \frac{6T_s - 4T^{\text{fus.}} - 2T_s(x^s(t)}{e_s} \tag{A.35}$$

$$\bar{\phi}_s^{bc}(t) = \lambda_s \frac{6T_s - 4T_s(x^s(t) - 2T^{\text{fus.}}}{e_s} \tag{A.36}$$

et on retrouve les équations (10) et (11) données dans la section 4.1.

Dans la plate-forme, d'autres modèles pour calculer les flux de chaleur sont utilisés. Dans [70], celui présenté ici a été démontré précis dans le cas 1D par rapport à une solution de référence donnée par la résolution par différences finies de la conduction dans le solide donnée par l'équation (A.29).

A.2.3 L'utilisation des modèles 0D couplés et problème partitionné bien posé

Le modèle 0D de conduction à résoudre consiste en un ensemble d'équations différentielles ordinaires (EDO) composé de :

— les équations 0D (A.17) et (A.19) de conservation de la masse pour les phases liquide et solide couplées par le débit de masse calculé par l'équation (A.22) du front de fusion plan.
— l'équation 0D (A.18) de conservation de l'énergie pour la phase liquide avec des lois de fermeture pour les frontières $\partial\Omega_l(t)$.
— l'équation 0D (A.20) de conservation de l'énergie pour la phase solide avec des lois de fermeture donnant les flux de chaleur $\bar{\phi}_s^{ls}(t)$ et $\bar{\phi}_s^{bc}(t)$.

Les lois de fermetures peuvent provenir de la résolution directe des équations dans le liquide et le solide (par exemple par différences finies) mais pas nécessairement (données par une hypothèse de profil quadratique de température dans le solide ou données sous la forme de corrélations de Nusselt-Rayleigh dans le domaine liquide (voir [12] ou [95] par exemple))

Ensuite, dans la plate-forme et notamment pour le couplage entre les domaines de la section 4.1, le système d'équations est divisé en plusieurs *blocs*, correspondant aux partitions, qui sont couplés et résolus sur la macro boucle en temps par le schéma de couplage. Néanmoins, le couplage entre ces blocs 0D *n'est pas libre* mais est *imposé* de sorte que le problème de décomposition de domaine sous-jacent soit bien posé et que les équations locales de

conservation soient bien couplées (par exemple un couplage de type Dirichlet/Neumann). *Le problème partitionné, par construction à partir des équations locales de conservation, est bien posé si et seulement si le problème de décomposition de domaine sous-jacent est bien posé.*

Bibliographie

[1] A. C. Aitken, *XXV.—On Bernoulli's Numerical Solution of Algebraic Equations*, Proceedings of the Royal Society of Edinburgh, 46 (1927), pp. 289–305, https://doi.org/10.1017/S0370164600022070, https://www.cambridge.org/core/journals/proceedings-of-the-royal-society-of-edinburgh/article/xxvon-bernoullis-numerical-solution-of-algebraic-equations/64D4A7C56F1EFEC696AF68D7870DB451 (accessed 2017-10-06).

[2] A. Alfonsi, C. Rabiti, D. Mandelli, J. Cogliati, R. Kinoshita, and A. Naviglio, *Dynamic event tree analysis through RAVEN*, in ANS PSA 2013 International Topical Meeting on Probabilistic Safety Assessment and Analysis, 2013.

[3] I. Antcheva, M. Ballintijn, B. Bellenot, M. Biskup, R. Brun, N. Buncic, P. Canal, D. Casadei, O. Couet, V. Fine, L. Franco, G. Ganis, A. Gheata, D. G. Maline, M. Goto, J. Iwaszkiewicz, A. Kreshuk, D. M. Segura, R. Maunder, L. Moneta, A. Naumann, E. Offermann, V. Onuchin, S. Panacek, F. Rademakers, P. Russo, and M. Tadel, *ROOT - a C++ framework for petabyte data storage, statistical analysis and visualization*, Computer Physics Communications, 180 (2009), pp. 2499–2512.

[4] Apache Software Foundation, *Apache Commons Math, Release 3.6.1*, 2016, http://commons.apache.org/math.

[5] F. Armero and J. C. Simo, *A new unconditionally stable fractional step method for non-linear coupled thermomechanical problems*, International Journal for Numerical Methods in Engineering, 35 (1992), pp. 737–766, https://doi.org/10.1002/nme.1620350408.

[6] S. Bajard, *Comparaison de la modélisation du bain de corium en coeur par MAAP4 / l'analyse scénario / PROCOR GEN III sur le cas NM3-LOOP650*, Tech. Report CEA/DEN/CAD/DTN/SMTA/LPMA/NT/2015-64/A, CEA Cadarache, 2015.

[7] A. Bellen and M. Zennaro, *The use of Runge-Kutta formulae in waveform relaxation methods*, Applied Numerical Mathematics, 11 (1993), pp. 95–114, https://doi.org/10.1016/0168-9274(93)90042-P, http://www.

sciencedirect.com/science/article/pii/016892749390042P (accessed 2016-11-03).

[8] P. Birken, T. Gleim, D. Kuhl, and A. Meister, *Fast solvers for unsteady thermal fluid structure interaction*, International Journal for Numerical Methods in Fluids, 79 (2015), pp. 16–29, https://doi.org/10.1002/fld.4040. fld.4040.

[9] P. Birken, K. J. Quint, S. Hartmann, and A. Meister, *On coupling schemes for heat transfer in FSI applications*, in Proceedings of the International Workshop on Fluid-Structure Interaction : Theory, Numerics and Applications Herrsching, 2009, pp. 21–30.

[10] J. Bloch, *Effective Java (2Nd Edition) (The Java Series)*, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2 ed., 2008.

[11] N. Boccara, *Modeling Complex Systems*, Graduate Texts in Physics, Springer New York, New York, NY, 2010, https://doi.org/10.1007/978-1-4419-6562-2.

[12] J. M. Bonnet and J. M. Seiler, *Thermohydraulic phenomena in corium pool : the BALI experiment*, in Proc. of ICONE 7, Tokyo, Japan, 1999.

[13] C. Brayer and G. Berthoud, *An Attempt to Model Stratified Thermal Explosions with the Multidimensional, Multicomponent Code {MC3D}*, in Multiphase Flow 1995, A. S. F. Bataille, ed., Elsevier, Amsterdam, 1995, pp. 483 – 495, http://www.sciencedirect.com/science/article/pii/B9780444818119500482.

[14] L. Carénini and F. Fichot, *The Impact of Transient Behaviour of Corium in the Lower Head of a Reactor Vesel for In-Vessel Melt Retention Strategies*, in In proc. of the 2016 24th INternational COnference on Nuclear Engineering ICONE24, Charlotte, North Carolina, 2016.

[15] L. Carénini and F. Fichot, *Definition of in-vessel retention (ivr) tests for codes benchmark*, tech. report, Institut de radioprotection et de sûreté nucléaire (IRSN), 2017.

[16] P. Causin, J. F. Gerbeau, and F. Nobile, *Added-mass effect in the design of partitioned algorithms for fluid–structure problems*, Computer Methods in Applied Mechanics and Engineering, 194 (2005), pp. 4506–4527, https://doi.org/10.1016/j.cma.2004.12.005, http://www.sciencedirect.com/science/article/pii/S0045782504005328 (accessed 2017-04-25).

[17] P. Chatelard, N. Reinke, S. Arndt, S. Belon, L. Cantrel, L. Carenini, K. Chevalier-Jabet, F. Cousin, J. Eckel, F. Jacq, C. Marchetto, C. Mun, and L. Piar, *ASTEC V2 severe accident integral code main features, current V2.0 modelling status, perspectives*, Nuclear Engineering and Design, 272 (2014), pp. 119–135,

https://doi.org/10.1016/j.nucengdes.2013.06.040, http://www.sciencedirect.com/science/article/pii/S0029549313005621 (accessed 2016-10-06).

[18] M. L. Crow and M. D. Ilić, *The Waveform Relaxation method for systems of differential/algebraic equations*, Mathematical and Computer Modelling, 19 (1994), pp. 67–84, https://doi.org/10.1016/0895-7177(94)90099-X, http://www.sciencedirect.com/science/article/pii/089571779490099X (accessed 2016-10-24).

[19] A. de Boer, A. van Zuijlen, and H. Bijl, *Comparison of conservative and consistent approaches for the coupling of non-matching meshes*, Computer Methods in Applied Mechanics and Engineering, 197 (2008), pp. 4284–4297, https://doi.org/10.1016/j.cma.2008.05.001, http://www.sciencedirect.com/science/article/pii/S0045782508001916.

[20] F. H. de Carvalho Junior and C. A. Rezende, *Performance evaluation of virtual execution environments for intensive computing on usual representations of multidimensional arrays*, Science of Computer Programming, 132, Part 1 (2016), pp. 29–49, https://doi.org/10.1016/j.scico.2016.04.005, http://www.sciencedirect.com/science/article/pii/S0167642316300065 (accessed 2017-04-21).

[21] J. Degroote, K.-J. Bathe, and J. Vierendeels, *Performance of a new partitioned procedure versus a monolithic procedure in fluid–structure interaction*, Computers & Structures, 87 (2009), pp. 793–801, https://doi.org/10.1016/j.compstruc.2008.11.013, http://linkinghub.elsevier.com/retrieve/pii/S0045794908002605 (accessed 2016-08-09).

[22] J. Degroote and J. Vierendeels, *Multi-level quasi-Newton coupling algorithms for the partitioned simulation of fluid–structure interaction*, Computer Methods in Applied Mechanics and Engineering, 225–228 (2012), pp. 14–27, https://doi.org/10.1016/j.cma.2012.03.010, http://www.sciencedirect.com/science/article/pii/S0045782512000862 (accessed 2017-03-22).

[23] C. Denis, P. de Oliveira Castro, and E. Petit, *Verificarlo : Checking floating point accuracy through Monte Carlo arithmetic*, in 23nd IEEE Symposium on Computer Arithmetic, ARITH 2016, Silicon Valley, CA, USA, July 10-13, 2016, 2016, pp. 55–62, https://doi.org/10.1109/ARITH.2016.31.

[24] V. Dolean, P. Jolivet, and F. Nataf, *An Introduction to Domain Decomposition Methods*, Other Titles in Applied Mathematics, Society for Industrial and Applied Mathematics, Nov. 2015, https://doi.org/10.1137/1.9781611974065.

[25] EDF, *Verrou : a floating-point rounding errors checker*, 2017, http://edf-hpc.github.io/verrou/vr-manual.html.

[26] H. Esmaili and M. Khatib-Rahbar, *Analysis of in-vessel retention and ex-vessel fuel coolant interaction for AP1000*, Tech. Report NUREG/CR-6849 ERI/NRC 04-201, U.S. Nuclear Regulatory Commission, Office of Nuclear Regulatory Research, 2004.

[27] C. Farhat and M. Lesoinne, *Two efficient staggered algorithms for the serial and parallel solution of three-dimensional nonlinear transient aeroelastic problems*, Computer Methods in Applied Mechanics and Engineering, 182 (2000), pp. 499 – 515, https://doi.org/10.1016/S0045-7825(99)00206-6, http://www.sciencedirect.com/science/article/pii/S0045782599002066.

[28] C. Farhat, M. Lesoinne, and N. Maman, *Mixed explicit/implicit time integration of coupled aeroelastic problems : Three-field formulation, geometric conservation and distributed solution*, International Journal for Numerical Methods in Fluids, 21 (1995), pp. 807–835, https://doi.org/10.1002/fld.1650211004/full.

[29] C. Farhat, K. C. Park, and Y. Dubois-Pelerin, *An unconditionally stable staggered algorithm for transient finite element analysis of coupled thermoelastic problems*, Computer methods in applied mechanics and engineering, 85 (1991), pp. 349–365, http://www.sciencedirect.com/science/article/pii/004578259190102C (accessed 2016-04-01).

[30] C. Farhat, A. Rallu, K. Wang, and T. Belytschko, *Robust and provably second-order explicit–explicit and implicit–explicit staggered time-integrators for highly non-linear compressible fluid–structure interaction problems*, International Journal for Numerical Methods in Engineering, 84 (2010), pp. 73–107, https://doi.org/10.1002/nme.2883.

[31] C. Farhat and N. Sobh, *A consistency analysis of a class of concurrent transient implicit/explicit algorithms*, Computer methods in applied mechanics and engineering, 84 (1990), pp. 147–162, http://www.sciencedirect.com/science/article/pii/0045782590901142 (accessed 2017-04-25).

[32] C. Farhat, K. G. van der Zee, and P. Geuzaine, *Provably second-order time-accurate loosely-coupled solution algorithms for transient nonlinear computational aeroelasticity*, Computer Methods in Applied Mechanics and Engineering, 195 (2006), pp. 1973–2001, https://doi.org/10.1016/j.cma.2004.11.031, http://linkinghub.elsevier.com/retrieve/pii/S0045782505001945 (accessed 2016-08-09).

[33] Fauske & Associates Inc., *MAAP4 - modular accident analysis program for LWR power plants*, in Code Structure and Theory, vol. 2, Electric Power Research Institute, 1994, ch. 1.

[34] Fauske & Associates Inc., *MAAP5 - modular accident analysis program for LWR power plants*, in Code Structure and Theory, vol. 2, Electric Power Research Institute, 2008, ch. 3.

[35] C. A. Felippa and K. C. Park, *Staggered transient analysis procedures for coupled mechanical systems : formulation*, Computer Methods in Applied Mechanics and Engineering, 24 (1980), pp. 61–111, http://www.sciencedirect.com/science/article/pii/0045782580900407 (accessed 2016-08-09).

[36] C. A. Felippa, K. C. Park, and C. Farhat, *Partitioned analysis of coupled mechanical systems*, Computer methods in applied mechanics and engineering, 190 (2001), pp. 3247–3270, http://www.sciencedirect.com/science/article/pii/S0045782500003911 (accessed 2016-08-09).

[37] L. Formaggia, J.-F. Gerbeau, F. Nobile, and A. Quarteroni, *Numerical Treatment of Defective Boundary Conditions for the Navier-Stokes Equations*, Research Report RR-4093, INRIA, 2001, https://hal.inria.fr/inria-00072539. Projet M3N.

[38] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns : Elements of Reusable Object-oriented Software*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

[39] M. J. Gander and E. Hairer, *Nonlinear Convergence Analysis for the Parareal Algorithm*, in Domain Decomposition Methods in Science and Engineering XVII, U. Langer, M. Discacciati, D. E. Keyes, O. B. Widlund, and W. Zulehner, eds., no. 60 in Lecture Notes in Computational Science and Engineering, Springer Berlin Heidelberg, 2008, pp. 45–56, https://doi.org/10.1007/978-3-540-75199-1_4. DOI :.

[40] M. J. Gander and S. Vandewalle, *On the Superlinear and Linear Convergence of the Parareal Algorithm*, in Domain Decomposition Methods in Science and Engineering XVI, O. B. Widlund and D. E. Keyes, eds., no. 55 in Lecture Notes in Computational Science and Engineering, Springer Berlin Heidelberg, 2007, pp. 291–298, https://doi.org/10.1007/978-3-540-34469-8_34. DOI :.

[41] V. Ganine, N. J. Hills, and B. L. Lapworth, *Nonlinear acceleration of coupled fluid–structure transient thermal problems by Anderson mixing*, International Journal for Numerical Methods in Fluids, 71 (2013), pp. 939–959, https://doi.org/10.1002/fld.3689.

[42] Gatzhammer, *Efficient and Flexible Partitioned Simulation of Fluid-Structure Interactions*, PhD thesis, Technische Universität München, 2014.

[43] F. Gaudier, *URANIE : the CEA/DEN uncertainty and sensitivity platform*, Procedia - Social and Behavioral Sciences, 2 (2010), pp. 7660–7661.

[44] J.-F. Gerbeau and M. Vidrascu, *A Quasi-Newton Algorithm Based on a Reduced Model for Fluid-Structure Interaction Problems in Blood Flows*, ESAIM : Mathematical Modelling and Numerical Analysis, 37 (2003), pp. 631–647, https://doi.org/10.1051/m2an:2003049.

[45] M. B. Giles, *Stability analysis of numerical interface conditions in fluid-structure thermal analysis*, International Journal for Numerical Methods in Fluids, 25 (1997), pp. 421–436, https://doi.org/10.1002/(SICI)1097-0363(19970830)25:4<421::AID-FLD557>3.0.CO;2-J.

[46] H. Guillard and C. Farhat, *On the significance of the geometric conservation law for flow computations on moving meshes*, Computer Methods in Applied Mechanics and Engineering, 190 (2000), pp. 1467–1482, https://doi.org/10.1016/S0045-7825(00)00173-0, http://www.sciencedirect.com/science/article/pii/S0045782500001730 (accessed 2016-08-09).

[47] V. Guillard, F. Fichot, P. Boudier, M. Parent, and R. Roser, *ICARE/CATHARE coupling : three-dimensional thermal-hydraulics of lwr severe accidents*, in Proceedings of ICONE 9, 9th International Conference on Nuclear Engineering, Nice Acropolis, France, 2001.

[48] G. Gundersen and T. Steihaug, *Data structures in Java for matrix computations*, Concurrency and Computation : Practice and Experience, 16 (2004), pp. 799–815, https://doi.org/10.1002/cpe.793.

[49] A. Hakobyan, T. Aldemir, R. Denning, S. Dunagan, D. Kunsman, B. Rutt, and U. Catalyurek, *Dynamic generation of accident progression event trees*, Nuclear Engineering and Design, 238 (2008), pp. 3457–3467, https://doi.org/10.1016/j.nucengdes.2008.08.005, http://www.sciencedirect.com/science/article/pii/S0029549308004470.

[50] M. Heil, *An efficient solver for the fully coupled solution of large-displacement fluid–structure interaction problems*, Computer Methods in Applied Mechanics and Engineering, 193 (2004), pp. 1–23, https://doi.org/10.1016/j.cma.2003.09.006, http://www.sciencedirect.com/science/article/pii/S0045782503005164.

[51] M. Heil, A. L. Hazel, and J. Boyle, *Solvers for large-displacement fluid–structure interaction problems : segregated versus monolithic approaches*, Computational Mechanics, 43 (2008), pp. 91–101, https://doi.org/10.1007/s00466-008-0270-6.

[52] K. J. in 't Hout, *On the convergence of waveform relaxation methods for stiff nonlinear ordinary differential equations*, Applied Numerical Mathematics, 18 (1995), pp. 175–190, https://doi.org/10.1016/0168-9274(95)00052-V, http://www.sciencedirect.com/science/article/pii/016892749500052V (accessed 2016-11-03).

[53] B. M. Irons and R. C. Tuck, *A version of the aitken accelerator for computer iteration*, International Journal for Numerical Methods in Engineering, 1 (1969), pp. 275–277, https://doi.org/10.1002/nme.1620010306, https://arxiv.org/abs/https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.1620010306.

[54] D. Jacquemain, *Les accidents de fusion du coeur des réacteurs nucléaires de puissance : état des connaissances (French)*, Collection sciences et techniques, EDP sciences, Les Ulis, 2013.

[55] J. K. M. Jansen, R. M. M. Mattheij, M. T. M. Penders, and W. H. A. Schilders, *Stability and efficiency of waveform relaxation methods*, Computers & Mathematics with Applications, 28 (1994), pp. 153–166, https://doi.org/10.1016/0898-1221(94)00103-0, http://www.sciencedirect.com/science/article/pii/0898122194001030 (accessed 2016-11-04).

[56] F. Jedrzejewski, *Introduction aux méthodes numériques*, Spinger, 2005.

[57] M. M. Joosten, W. G. Dettmer, and D. Perić, *Analysis of the block Gauss-Seidel solution procedure for a strongly coupled model problem with reference to fluid-structure interaction*, International Journal for Numerical Methods in Engineering, 78 (2009), pp. 757–778, https://doi.org/10.1002/nme.2503.

[58] JUnit, *JUnit test tool, http ://www.junit.org*, 2017, http://junit.org/junit4/ (accessed 2017-07-07).

[59] D. R. Karanki, T.-W. Kim, and V. N. Dang, *A dynamic event tree informed approach to probabilistic accident sequence modeling : Dynamics and variabilities in medium LOCA*, Reliability Engineering & System Safety, 142 (2015), pp. 78–91, https://doi.org/10.1016/j.ress.2015.04.011, http://www.sciencedirect.com/science/article/pii/S0951832015001234.

[60] C. Kassiotis, *Nonlinear fluid-structure interaction : a partitioned approach and its application through component technology*, phdthesis, Université Paris-Est, Nov. 2009, https://pastel.archives-ouvertes.fr/tel-00453394/document (accessed 2017-04-18).

[61] C. Kassiotis, A. Ibrahimbegovic, R. Niekamp, and H. G. Matthies, *Nonlinear fluid–structure interaction problem. Part I : implicit partitioned algorithm, nonlinear stability proof and validation examples*, Computational Mechanics, 47 (2011), pp. 305–323, https://doi.org/10.1007/s00466-010-0545-6.

[62] D. E. Keyes, L. C. McInnes, C. Woodward, W. Gropp, E. Myra, M. Pernice, J. Bell, J. Brown, A. Clo, J. Connors, E. Constantinescu, D. Estep, K. Evans, C. Farhat, A. Hakim, G. Hammond, G. Hansen, J. Hill, T. Isaac, X. Jiao, K. Jordan, D. Kaushik, E. Kaxiras, A. Koniges, K. Lee, A. Lott, Q. Lu, J. Magerlein, R. Maxwell, M. McCourt, M. Mehl, R. Pawlowski, A. P. Randles, D. Reynolds, B. Rivière, U. Rüde, T. Scheibe, J. Shadid, B. Sheehan, M. Shephard, A. Siegel, B. Smith, X. Tang, C. Wilson, and B. Wohlmuth, *Multiphysics simulations : Challenges and opportunities*, International Journal of High Performance Computing Applications, 27

(2013), pp. 4–83, https://doi.org/10.1177/1094342012468181, http://hpc.sagepub.com/content/27/1/4.abstract.

[63] S. Kondo, Y. Tobita, K. Morita, and N. Shirakawa, *Simmer-*III *: an advanced computer program for lmfbr severe accident analysis*, in International conference on design and safety of advanced nuclear power plants, ANP 92, Tokyo, Japan, 1992.

[64] J. J. Kreeft, M. Weghs, A. H. Van Zuijlen, and H. Bijl, *Multi-level and quasi-Newton acceleration for strongly coupled partitioned fluid-structure interaction*, in Coupled Problems 2011 : Proceedings of the 4th International Conference on Computational Methods for Coupled Problems in Science and Engineering, Kos, Greece, 20-22 June 2011, CIMNE, 2011, http://repository.tudelft.nl/view/ir/uuid:7a6d39d8-482c-4aac-a319-86216d29b057/ (accessed 2016-11-29).

[65] S. Kudriakov, F. Dabbene, E. Studer, A. Beccantini, J. P. Magnaud, H. Paillère, A. Bentaib, A. Bleyer, J. Malet, E. Porcheron, and C. Caroli, *The TONUS CFD code for hydrogen risk analysis : Physical models, numerical schemes and validation matrix*, Nuclear Engineering and Design, 238 (2008), pp. 551–565, https://doi.org/10.1016/j.nucengdes.2007.02.048, http://www.sciencedirect.com/science/article/pii/S0029549307003457 (accessed 2016-10-06).

[66] U. Küttler and W. A. Wall, *Fixed-point fluid–structure interaction solvers with dynamic relaxation*, Computational Mechanics, 43 (2008), pp. 61–72, https://doi.org/10.1007/s00466-008-0255-5.

[67] R. Le Tellier and L. Saas, *Fourniture de résultats de calcul procor dans le cadre du wp2.2 du projet ivmr*, tech. report, CEA Cadarache DEN/DTN/SMTA/LPMA, 2017.

[68] R. Le Tellier, L. Saas, and S. Bajard, *Transient stratification modelling of a corium pool in a LWR vessel lower head*, Nuclear Engineering and Design, 287 (2015), pp. 68–77.

[69] R. Le Tellier, L. Saas, and F. Payot, *Phenomenological analyses of corium propagation in LWRs : the PROCOR software platform*, in Proc. of the 7th European Review Meeting on Severe Accident Research ERMSAR-2015, Marseille, France, 2015.

[70] R. Le Tellier, E. Skrzypek, and L. Saas, *On the treatment of plane fusion front in lumped parameter thermal models with convection*, Applied Thermal Engineering, 120 (2017), pp. 314–326, https://doi.org/10.1016/j.applthermaleng.2017.03.108, http://www.sciencedirect.com/science/article/pii/S1359431116327119 (accessed 2017-04-10).

[71] J. Liu and Y.-L. Jiang, *A parareal algorithm based on waveform relaxation*, Mathematics and Computers in Simulation, 82 (2012), pp. 2167–2181, https://doi.org/10.1016/j.matcom.2012.05.017, http://www.sciencedirect.com/science/article/pii/S0378475412001358 (accessed 2016-10-26).

[72] H. Lundvall, P. Fritzson, and B. Bachmann, *Event handling in the openmodelica compiler and runtime system*, Linköping University Electronic Press, 2008.

[73] Y. Maday and G. Turinici, *The Parareal in Time Iterative Solver : a Further Direction to Parallel Implementation*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 441–448, https://doi.org/10.1007/3-540-26825-1_45.

[74] G. Mao and L. R. Petzold, *Efficient integration over discontinuities for differential-algebraic systems*, Computers & Mathematics with Applications, 43 (2002), pp. 65–79, https://doi.org/10.1016/S0898-1221(01)00272-3, http://www.sciencedirect.com/science/article/pii/S0898122101002723 (accessed 2017-06-21).

[75] M. Mehl, B. Uekermann, H. Bijl, D. Blom, B. Gatzhammer, and A. van Zuijlen, *Parallel coupling numerics for partitioned fluid–structure interaction simulations*, Computers & Mathematics with Applications, 71 (2016), pp. 869–891, https://doi.org/10.1016/j.camwa.2015.12.025, http://linkinghub.elsevier.com/retrieve/pii/S0898122115005933 (accessed 2016-08-09).

[76] R. Meignen, B. Raverdy, M. Buck, G. Pohlner, P. Kudinov, W. Ma, C. Brayer, P. Piluso, S.-W. Hong, M. Leskovar, M. Uršič, G. Albrecht, I. Lindholm, and I. Ivanov, *Status of steam explosion understanding and modelling*, Annals of Nuclear Energy, 74 (2014), pp. 125–133, https://doi.org/10.1016/j.anucene.2014.07.008, http://www.sciencedirect.com/science/article/pii/S0306454914003399. ERMSAR 2013 conference of the SARNET network.

[77] C. Michler, E. H. van Brummelen, and R. de Borst, *An interface Newton-Krylov solver for fluid-structure interaction*, International Journal for Numerical Methods in Fluids, 47 (2005), pp. 1189–1195, https://doi.org/10.1002/fld.850.

[78] S. Minami and S. Yoshimura, *Performance evaluation of nonlinear algorithms with line-search for partitioned coupling techniques for fluid-structure interactions*, International Journal for Numerical Methods in Fluids, 64 (2010), pp. 1129–1147, https://doi.org/10.1002/fld.2274.

[79] J. E. Moreira, S. P. Midkiff, and M. Gupta, *Supporting multidimensional arrays in Java*, Concurrency and Computation : Practice and Experience, 15 (2003), pp. 317–340, https://doi.org/10.1002/cpe.661.

[80] M. P., D. A., D. E., A. E., G. J., B. A., L. L., and C. C., *"coupling methodology within the software platform alliances"*, in Proc. of Int. Conf. on Computational Methods for Coupled Problems in Science and Engineering, 2005.

[81] T. Park and P. I. Barton, *State event location in differential-algebraic models*, ACM Trans. Model. Comput. Simul., 6 (1996), pp. 137–165, https://doi.org/10.1145/232807.232809.

[82] L. Petzold, *Differential/Algebraic Equations are not ODE's*, SIAM Journal on Scientific and Statistical Computing, 3 (1982), pp. 367–384, https://doi.org/10.1137/0903023.

[83] S. Piperno and C. Farhat, *Partitioned procedures for the transient solution of coupled aeroelastic problems–Part II : energy transfer analysis and three-dimensional applications*, Computer methods in applied mechanics and engineering, 190 (2001), pp. 3147–3170, http://www.sciencedirect.com/science/article/pii/S0045782500003868 (accessed 2016-08-09).

[84] S. Piperno, C. Farhat, and B. Larrouturou, *Partitioned procedures for the transient solution of coupled aroelastic problems Part I : Model problem, theory and two-dimensional application*, Computer Methods in Applied Mechanics and Engineering, 124 (1995), pp. 79 – 112, https://doi.org/10.1016/0045-7825(95)92707-9, http://www.sciencedirect.com/science/article/pii/0045782595927079.

[85] J. Printz, *Architecture logicielle : Concevoir des applications simples, sûres et adaptables*, Dunod, 2006.

[86] I. Ramière and T. Helfer, *Iterative residual-based vector methods to accelerate fixed point iterations*, Computers & Mathematics with Applications, 70 (2015), pp. 2210–2226, https://doi.org/10.1016/j.camwa.2015.08.025, http://linkinghub.elsevier.com/retrieve/pii/S0898122115004046 (accessed 2017-01-16).

[87] B. Schrefler, L. Simoni, and E. Turska, *Standard staggered and staggered Newton schemes in thermo-hydro-mechanical problems*, Computer Methods in Applied Mechanics and Engineering, 144 (1997), pp. 93–109, https://doi.org/10.1016/S0045-7825(96)01170-X, http://linkinghub.elsevier.com/retrieve/pii/S004578259601170X (accessed 2017-04-25).

[88] B. R. Sehgal, *Nuclear safety in Light Water Reactors : Severe Accident Phenomenology*, Elsevier/Academic Press, Amsterdam ; Boston, 1st ed ed., 2012.

[89] M. Sonnenkalb, J. Peschke, M. Kloos, and B. Krzycacz-Hausmann, *MCDET and MELCOR : An Example of a Stochastic Module coupled with an Integral Code for PSA Level 2*, in International Workshop on Level 2 PSA and Severe Accident Management, 2009.

[90] B. Spindler, B. Tourniaire, and J. M. Seiler, *Simulation of MCCI with the TOLBIAC-ICB code based on the phase segregation model*, Nuclear Engineering and Design, 236 (2006), pp. 2264–2270.

[91] M. A. Storti, N. M. Nigro, R. R. Paz, and L. D. Dalcín, *Strong coupling strategy for fluid–structure interaction problems in supersonic regime via fixed point iteration*, Journal of Sound and Vibration, 320 (2009), pp. 859–877, http://www.sciencedirect.com/science/article/pii/S0022460X08007542 (accessed 2017-04-25).

[92] B. Sundman, U. R. Kattner, M. Palumbo, and S. G. Fries, *Opencalphad - a free thermodynamic software*, Integrating Materials and Manufacturing Innovation, 4 (2015), p. 1, https://doi.org/10.1186/s40192-014-0029-1.

[93] B. Tourniaire, B. Spindler, G. Ratel, J. M. Seiler, B. Iooss, M. Marquès, F. Gaudier, and G. Greffier, *"the LEONAR code : a new tool for PSA level 2 Analyses"*, in Proc. of Joint OECD/NEA - EC/SARNET2 Workshop on In-Vessel Coolability, vol. NEA/CSNI/R(2010)11, p. 105-121, 2009.

[94] J. Vierendeels, L. Lanoye, J. Degroote, and P. Verdonck, *Implicit coupling of partitioned fluid–structure interaction problems with reduced order models*, Computers & Structures, 85 (2007), pp. 970–976, http://www.sciencedirect.com/science/article/pii/S0045794906003865 (accessed 2017-05-05).

[95] L. Zhang, Y. Zhou, Y. Zhang, W. Tian, S. Qiu, and G. Su, *Natural convection heat transfer in corium pools : A review work of experimental studies*, Progress in Nuclear Energy, 79 (2015), pp. 167–181, https://doi.org/10.1016/j.pnucene.2014.11.021, http://www.sciencedirect.com/science/article/pii/S014919701400328X (accessed 2016-10-06).

Titre : Couplage et synchronisation de modèles dans un code scénario d'accidents graves dans les réacteurs nucléaires.

Mots clés : schéma de couplage, synchronisation, approche partitionnée, plate-forme de couplage, accident grave dans les réacteurs nucléaires.

Résumé : Le travail de thèse détaillé dans ce manuscrit présente la résolution numérique *partitionnée* et *synchronisée* des *systèmes complexes* survenant lors de la simulation d'accidents graves dans les réacteurs à eau légère au sein de la plate-forme logicielle CEA PROCOR.
La simulation d'accidents graves se fait par la résolution de systèmes complexes constitués des différents phénomènes physiques couplés apparaissant dans le réacteur nucléaire. Les phénomènes sont multi-physiques, par exemple la neutronique, la thermo-hydraulique, la thermochimie, la mécanique, etc., et multi-échelles avec des temps caractéristiques allant de la microseconde à l'année, et des masses allant du gramme à la centaine de tonnes. De plus, le manque de de données et de connaissances phénoménologiques font que la modélisation des différents phénomènes, encore incertaine, est en constante évolution. Enfin, les modèles à états du système peuvent déclencher des événements de changement d'état associés à des discontinuités.
Permettant de traiter chaque modèle avec une résolution numérique adaptée au phénomène sous-jacent et offrant modularité et évolutivité des modèles, l'approche partitionnée de résolution des systèmes complexes est par conséquent particulièrement adaptée au contexte des accidents graves. Cette approche propose des algorithmes partitionnés de résolution des problèmes couplés associés aux systèmes complexes permettant, en faisant le bon choix de l'algorithme, de retrouver la précision d'une résolution monolithique totalement couplée.
Dans ce travail de thèse, un formalisme de représentation des systèmes complexes a été proposé. Les algorithmes partitionnés ont été étudiés et adaptés pour la résolution de ces systèmes. Un algorithme de synchronisation sur les événements des modèles du système a été proposé. Une architecture logicielle de couplage, dans laquelle ont été intégrés les différents algorithmes de partitionnement et de synchronisation, a été ajoutée à la plate-forme PROCOR. Des résultats numériques sur des couplages d'importances pour les accidents graves et sur des cas industriels valident et montrent le potentiel des algorithmes implémentés. Cette approche et les résultats associés sont généraux et pourraient s'appliquer à d'autres domaines.

Title : Coupling and synchronisation of models in a code for severe accidents in nuclear reactors.

Keywords : coupling scheme, synchronisation, partitioned approach, coupling software, severe accident in nuclear power plants.

Abstract : This Ph.D topic is focused on the *partitioned* and *synchronised* numerical solving of *complex systems* arising in the simulation of severe accidents in light water reactors by the CEA PROCOR platform.
The simulation of severe accidents is made possible by the solving of complex systems composed of the various coupled physical phenomena appearing inside the nuclear reactor. Phenomena are multi physics, such as neutronic, thermal hydraulic, thermochemistry, mechanic, etc., and multi scales, with characteristic times going from microseconds to years and masses going from grams to hundred of tons. Furthermore, because of the lack of physical data and phenomenological knowledge the modeling of the various phenomena is still uncertain and in constant development. Finally, models of the system can trigger event corresponding to internal change of state which are often associated with strong discontinuities.
Allowing to handle each model with a numerical scheme adapted to the underlying physical phenomenon and offering modularity and evolutivity, the partitioned approach of solving the complex systems is particularly suitable in the context of severe accidents. This approach offers many partitioned algorithms to solve the coupled problems associated to the complex systems and, with the appropriate choice of algorithm, the precision of the monolithic solving can even be retrieved.
In this work, a new formalism to represent the complex systems is described. The partitioned algorithms are studied and adapted to the solving of these systems. A new synchronisation algorithm is proposed and allow the coupled models to be synchronised on their internal events. Furthermore, an coupling software architecture, in which the previous algorithms have been implemented, has been added to already existing PROCOR industrial platform. Numerical results on important couplings for severe accidents and from industrial applications of the PROCOR software are given and allow to validate the implemented algorithm and show the promising potential of the coupling environment.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France