Livrable L1.3.3a : Démonstrateur de couplage fluide structure EDF

IOLS/WP1.3/RAP/démonstrateur EDF

	Nom	Date	Visa
Auteur (s) :	**C. Caremoli A. Ribes E. Fayolle F. Jusserand N. Greffet E. Longatte Lacazedieu**	**12/7/2007**	
Vérificateur :	**V. Lefebvre**	**12/7/2007**	
Approbateur :	**E. Lorentz**	**03/09/2007**	

Table des matières

1. INTRODUCTION 3
1.1 Objet du document 3
1.2 Contexte et objectifs 3
2. CAS D'UTILISATION 5
3. LES CAS DE CALCUL 7
3.1 Tube mobile dans un faisceau de tubes en écoulement transverse 7
3.2 Tuyau d'arrosage 7
3.3 Lame flexible 8
4. LES CODES DE CALCUL DU COUPLAGE 10
4.1 Code_Aster 10
4.2 Code_Saturne 10
5. LE COUPLAGE 11
5.1 Échange de données 11
5.2 Calcul transitoire et algorithme de couplage 12
6. LE COUPLAGE INFORMATIQUE 13
6.1 Les outils de couplage de YACS 13
6.1.1 Les caractéristiques des composants 13
6.1.2 L'assemblage des composants 15
6.2 Le composant Saturne 15
6.3 Le composant Aster 16
6.4 Le composant SATAST 16
6.5 Le composant ASTSAT 17
6.6 Le composant Coupler 18
6.7 L'assemblage des composants pour le couplage 19
7. LES RESULTATS 21
7.1 Le faisceau 21
7.2 Le tuyau 23
7.3 La lame flexible 24
8. REALISATION INFORMATIQUE 27
8.1 Le module COSMETHYC 27
8.2 Les composants SATAST, ASTSAT et COUPLER 27
8.2.1 La définition des services SALOME 28
8.2.2 La création des ports datastream 28
8.2.3 L'utilisation des ports datastream 29

Table des matières

8.2.4 Traitement des erreurs et sortie 30
8.2.5 L'implémentation du service run 30
8.2.6 La déclaration du composant dans le fichier de couplage 31
8.2.7 Les composants ASTSAT et COUPLER 31
8.3 Les composants SATURNE 31
8.3.1 Le composant 32
8.3.2 Les services 32
8.3.3 Les ports datastream 32
8.3.4 L'implémentation 32
8.3.5 Déclaration dans le fichier de couplage 32
8.4 Le composant ASTER 33
8.4.1 Le composant 33
8.4.2 Les services du composant 33
8.4.3 Les ports datastream 34
8.4.4 L'implémentation du composant 34
8.4.5 Traitement des erreurs et sortie 34
8.4.6 Déclaration dans le fichier de couplage 34
8.5 Les couplages 35
9. CONCLUSION 36

Contacts

Contacts

Coordinateur Work-Package	**Vincent Lefebvre** EDF/RD/SINETICS ☎ ✍ vincent.lefebvre@edf.fr
Auteur	**Christian Caremoli** EDF/RD/SINETICS ☎ 0147654769 ✍ christian.caremoli@edf.fr
Auteur	**André Ribes** EDF/RD/SINETICS ☎ ✍ andre.ribes@edf.fr
Auteur	**Eric Fayolle** EDF/RD/SINETICS ☎ ✍ eric.fayolle@edf.fr
Auteur	**François Jusserand** EDF/RD/MFEE ☎ ✍ francois.jusserand@edf.fr
Auteur	**Nicolas Greffet** EDF/RD/AMA ☎ ✍ nicolas.greffet@edf.fr
Auteur	**Elisabeth Longatte-Lacazedieu** EDF/RD/MFEE ☎ ✍ elisabeth.longatte@edf.fr

Suivi des modifications

Suivi des modifications

Version/Révision		Références		Description des modifications	Auteur(s)
Indice	Date	Page	N° §		
1.0	2/07/07			*Version initiale*	CAREMOLI C.
2.0	9/07/07			*Suite à remarques diverses*	CAREMOLI C.
3.0	10/07/07			*pages 11et 12*	CAREMOLI C.
4.0	12/07/07			*relecture O. Morvant*	CAREMOLI C.

Liste de diffusion

Liste de diffusion Projet

N. Crouzet	**CEA**
A. Geay	**CEA**
L. Dada	**CEA**
P. Pasquet	**SAMTECH**
Y. Fricaud	**OpenCascade**
M. Kazakov	**OpenCascade**
J-Y Berthou	**EDF**
J. Bonelle	**EDF**
Y Fournier	**EDF**
S. Potapov	**EDF**
C. Durand	**EDF**
N. Tardieu	**EDF**
D. Thai-Van	**EDF**
F De-Vuyst	**ECP**
C. Saguez	**ECP**
J. Ryan	**ONERA**
A. Thomasset	**CS**

Liste de diffusion Externe

Fe Waeckel	**EDF**
E. Lorentz	**EDF**
F. Waeckel	**EDF**
E. Brière	**EDF**
F. Archambeau	**EDF**
D. Beaucourt	**EDF**

Documents associés

Documents Applicables

DA 1 : Convention

DA 2 : Annexe technique

Documents de Référence

DR 1 : Plate-forme SALOME : http://www.salome-platform.org

DR 2 : The Common Component Architecture (CCA) Forum : http://www.cca-forum.org

DR 3 : Le coupleur CALCIUM : http://www.irisa.fr/orap/Publications/Forum8/berthou.pdf

DR 4 : Le coupleur PALM : http://www.cerfacs.fr/globc/PALM_WEB/

DR 5 : Sébastien Lacour. Contribution à l'automatisation du déploiement d'applications sur des grilles de calcul. Thèse de doctorat, Université de Rennes 1, IRISA, Rennes, France, décembre 2005.

DR 6 : André Ribes. Contribution à la conception d'un modèle de programmation parallèle et distribué et sa mise en oeuvre au sein de plates-formes orientées objet et composant. Thèse de doctorat, Université de Rennes 1, IRISA, Rennes, France, décembre 2004.

DR 7 : Modèle d'échange de données MED : http://www.salome-platform.org/ex/doc2.2.8/med_fichier/doc/index.html

DR 8 : M.A. Fernandez Varela. Modèles simplifiés d'interaction fluide-structure. thèse de doctorat, Université Paris IX Dauphine, 2001.

DR 9 : J.Y. Renou. Une méthode eulerienne pour le calcul numérique de forces fluides-´elastiques.thèse de doctorat, Université Paris VI, 1998.

DR 10 : S. Turek, J. Hron : Proposal for numerical benchmarking of fluid-structure interaction between an elastic object and laminar incompressible flow in Technical report, FB Mathematik, Universitat Dortmund, 2006. Ergebnisberichte des Instituts fur Angewandte Mathematik, Nummer 312.

DR 11 : C. Caremoli et al. : IOLS/WP1.3/RAP/spécifications version 1.1

DR 12 : F. Huvelin, E. Longatte-Lacazedieu : Simulation numérique par couplage de codes des vibrations de structures induites par écoulements, rapport EDF H-I84-2006-01601-FR

Documents associés

DR 13 : Farhat C., Lesoinne M., Maman N., 1995. Mixed explicit/implicit time integration of coupled aeroelastic problems : three field formulation, geometric conservation and distribution solution, International Journal for Numerical Methods in Fluids. 21, 807-835.

DR 14 : Farhat C., Lesoinne M., 1997. Improved Staggered Algorithms for Serial and Parallel Solution of Three-Dimensional Nonlinear Transient Aeroelastic Problems. Center for Aerospace Structures- 97-11, University of Colorado, Boulder, Colorado, AIAA Journal.

DR 15 : Thomas P.D., Lombard C.K., 1979. Geometric Conservation Law and Its Application to Flow Computations on Moving Grids. AIAA Journal, 17, 1030 :1037.

DR 16 : Abouri D., Parry A., Hamdouni A., Longatte E. 2005. A stable fluid structure interaction algorithm : application to industrial problems, JPVT-05-1050-003604JPV.

Chapitre 1

1. Introduction

1.1 Objet du document

Ce document a pour objectif de décrire le démonstrateur de couplage EDF, construit sur la base du prototype de coupleur développé dans le WP1.3 du projet IOLS. Il met en œuvre les codes de calcul EDF *Code_Aster* et *Code_Saturne* pour modéliser une interaction fluide-structure.

Ce coupleur nommé YACS (dYnamic pArallel Coupling Supervisor) a été développé sur la base des spécifications écrites au début du projet [DR11].

Ce document peut être utilisé comme une introduction à la construction de couplages dans YACS.

Il s'agit du livrable L1.3.3a du Working Package 1.3 [DA2].

1.2 Contexte et objectifs

L'objectif du démonstrateur EDF de couplage fluide structure est de montrer que la synthèse du modèle de couplage initial de la plate-forme SALOME (version 3.X) et du modèle de couplage à base de ports "datastream" (utilisé par les outils de couplage PALM [DR4] et CALCIUM [DR3]) est effective dans le coupleur YACS développé dans le WP1.3. Cette démonstration doit être faite sur l'exemple du couplage *Code_Aster*/ *Code_Saturne* déjà réalisé avec l'outil CALCIUM.

Cette volonté de disposer d'un modèle de couplage enrichi part du constat suivant : le modèle de couplage SALOME demande souvent une restructuration difficile à réaliser pour certains codes existants. L'outil CALCIUM permet de réaliser des couplages entre codes existants sans nécessiter une restructuration profonde de ces codes. Une première intégration de CALCIUM dans SALOME avait été réalisée mais elle consistait en une cohabitation des deux modèles qui ne permettait pas de faire bénéficier chacun des apports de l'autre. En particulier, SALOME dispose du modèle MED [DR7] pour les maillages et champs qui n'était pas utilisable dans CALCIUM.

Le nouveau modèle de couplage proposé enrichit le modèle SALOME avec un mécanisme de connexion dynamique d'interface semblable à celui préconisé par le CCA Forum [DR2]. Les mécanismes de couplage par ports "datastream" sont ensuite construits au-dessus.

Les perspectives ouvertes par ce nouveau modèle sont nombreuses :

- couplage dans SALOME par ports "datastream" comme dans CALCIUM ou PALM
- extension des ports "datastream" vers les données de plus haut niveau comme le modèle MED
- parallélisation de ces mécanismes de couplage par utilisation du middleware PACO++ [DR6]
- construction de couplages à des niveaux d'abstraction plus élevés par utilisation de la mécanique de couplage d'interface

Le démonstrateur décrit dans ce rapport ne traite que du premier point et uniquement du mode CALCIUM.

Chapitre 1

2. Cas d'utilisation

Le cas d'utilisation retenu pour réaliser le démonstrateur de couplage EDF est le traitement de l'interaction fluide structure avec les outils *Code_Aster* et *Code_Saturne*. Le schéma de principe général du couplage est présenté ci-dessous. Il commence, en général, par un calcul statique suivi d'un calcul transitoire. Pour le démonstrateur, on ne s'intéressera qu'à la partie transitoire.

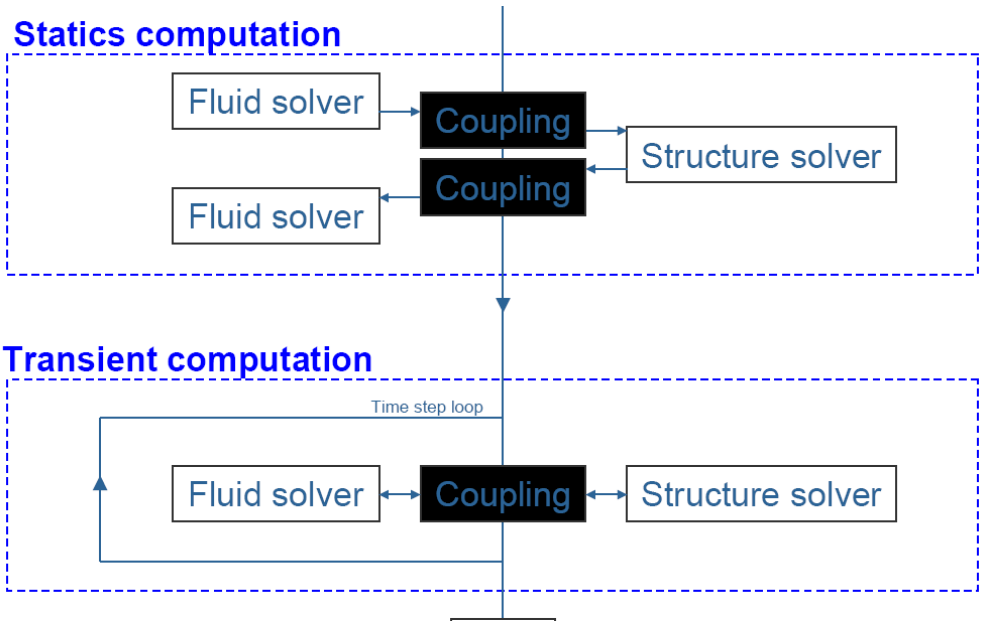

Figure 1 : Schéma de principe du couplage

Les échanges de données concernent des champs de déplacement, calculés par *Code_Aster,* et des champs de forces, calculés par *Code_Saturne*. Pour le démonstrateur, les hypothèses retenues pour le calcul de l'écoulement du fluide sont les suivantes : écoulement laminaire avec maillage mobile pris en charge par une méthode de type ALE (Arbitrary Lagrangian Eulerian). On travaillera également avec une hypothèse de petits déplacements pour la structure.

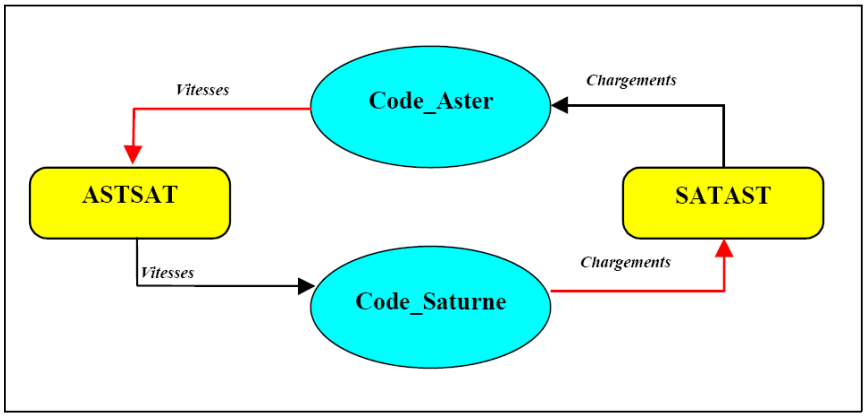

Figure 2 : Schéma d'échange et d'interpolation avec maillages incompatibles

Les champs échangés entre *Code_Saturne* et *Code_Aster* peuvent être de mêmes dimensions : 3D-3D ou de dimensions différentes : 3D-1D, par exemple. Les maillages sont en général incompatibles et sont donc interpolés au moyen des modules d'interpolation existants : SATAST et ASTSAT (figure 2). Les données identifiées par le nom "Vitesses" sur cette figure correspondent aux déplacements et vitesses de la structure. Les données identifiées par le nom "Chargements" correspondent aux forces induites par le champ de pression du fluide.

L'algorithme général de couplage de la phase transitoire est représenté ci-dessous.

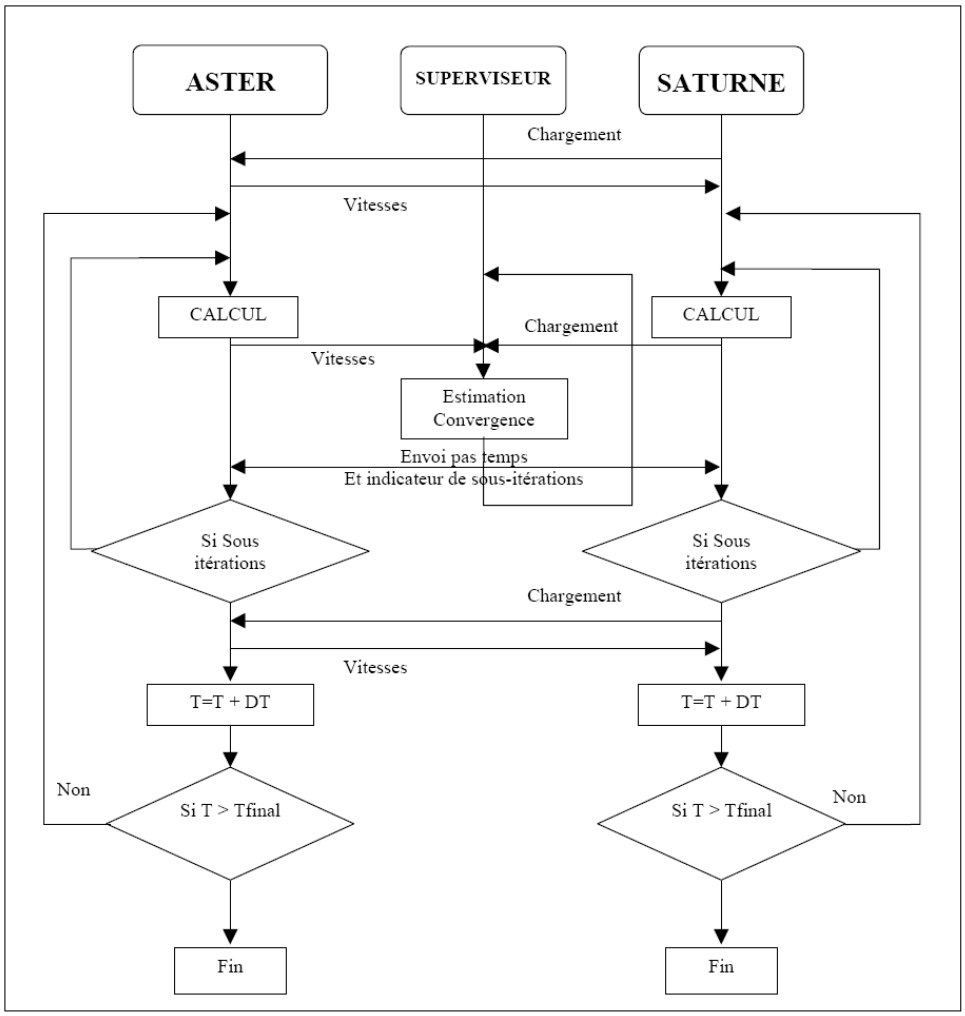

Figure 3 : Algorithme général de couplage de la phase transitoire

Le principe du couplage est basé sur une procédure itérative : chaque code avance dans le temps d'un pas de temps et envoie à la fin de ce pas de temps ses champs calculés à l'autre code. Il est possible de réaliser des sous itérations à chaque pas de temps pour converger avec plus de précision sur les champs de déplacement et de forces. Cette possibilité n'a pas été implémentée dans le démonstrateur mais le sera très prochainement car elle est indispensable pour garantir la stabilité du calcul dans de nombreux cas.

3. Les cas de calcul

Trois cas de calcul ont été retenus pour la mise en œuvre du démonstrateur de couplage : le cas d'un faisceau de tubes soumis à un écoulement transverse, le cas d'un écoulement interne dans une structure souple, type « tuyau d'arrosage », et le cas d'une structure souple soumise à un écoulement externe, type « lame flexible ». Ces trois cas sont brièvement décrits ci-dessous. On donne des références bibliographiques pour ceux qui souhaitent plus de détails.

3.1 Tube mobile dans un faisceau de tubes en écoulement transverse

L'étude des vibrations d'un faisceau de tubes peut être très exigeante en terme de taille de maillage à utiliser, en raison des contraintes induites par l'identification des chargements thermohydrauliques pariétaux. On se limitera donc pour notre étude à une cellule de 9 tubes, le faisceau complet étant obtenu avec des conditions aux limites périodiques. Seul le tube central est mobile. Tous ses voisins sont fixes. Le débit en entrée de la cellule est imposé et partout ailleurs, on utilise des conditions de périodicité. On dispose pour cette configuration de données expérimentales et numériques de référence.

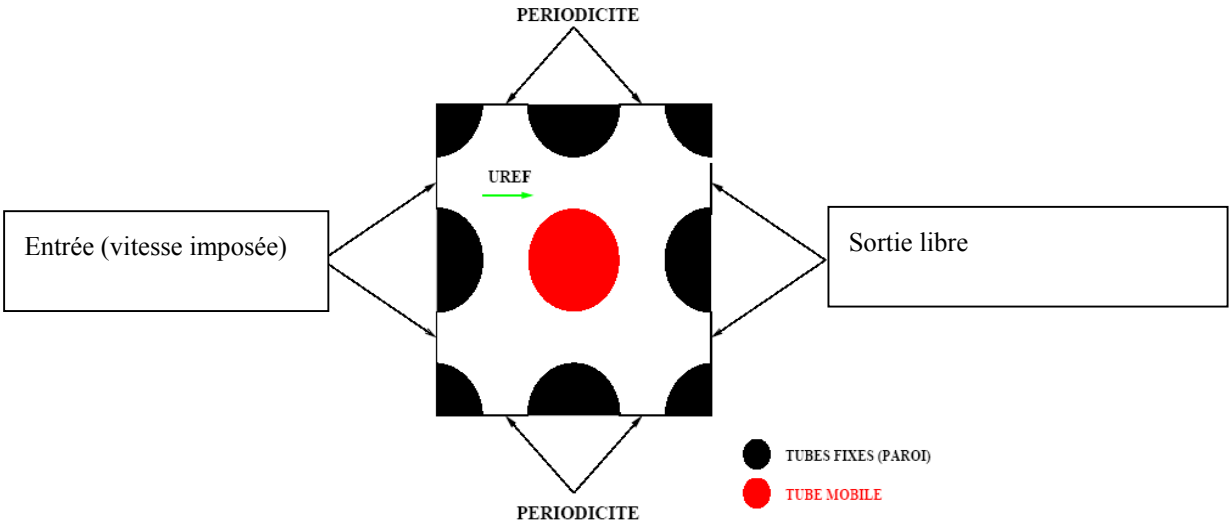

Figure 4 : Cellule de 9 tubes dont un mobile

Les dimensions retenues pour le calcul couplé sont :

Diamètre du tube : 0.01215 m

Pas du faisceau : 0.017496 m

3.2 Tuyau d'arrosage

Le cas d'un écoulement à l'intérieur d'un tuyau permet de tester le couplage pour une structure flexible. Ce cas présente un intérêt du point de vue de la validation physique car il permet de vérifier la capacité de l'outil à reproduire le transfert d'énergie à l'interface entre

les deux systèmes couplés, c'est-à-dire à simuler le transfert d'amortissement positif ou négatif d'un système vers l'autre et à reproduire le départ en instabilité de la structure sous l'effet du couplage avec l'écoulement. Il a été étudié et des simulations numériques ont été réalisées [DR8, DR9]. La géométrie utilisée avec ses dimensions caractéristiques est donnée ci-dessous. Ce cas est doté de données analytiques et numériques de référence.

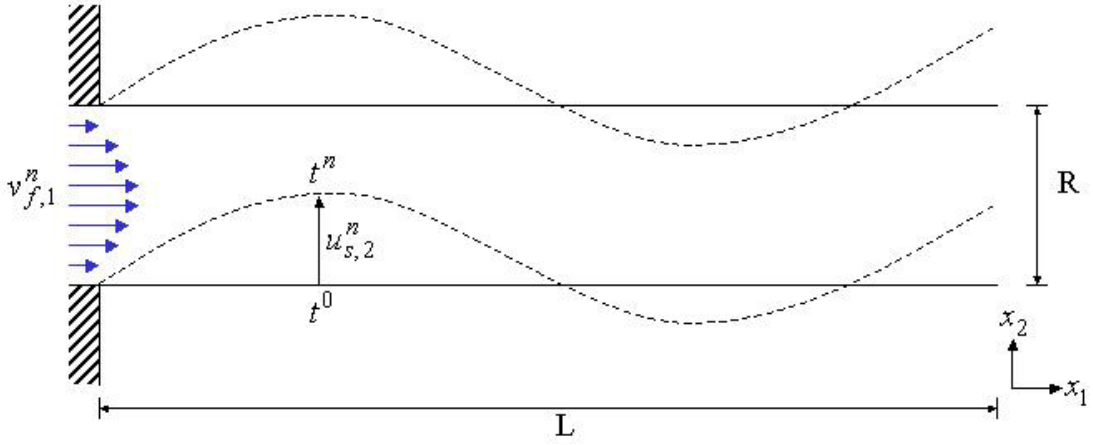

Figure 5 : Géométrie du tuyau d'arrosage

La longueur du tuyau est de 1 m. Son rayon interne est de 0.02 m et son épaisseur de 0.0004 m.

3.3 Lame flexible

Par rapport au cas précédent, le cas de la lame flexible derrière un cylindre est également un cas fortement étudié. Il présente l'intérêt de faire intervenir un écoulement ainsi qu'un comportement de structure flexible potentiellement non linéaire plus complexes. Ce cas de calcul est un benchmark [DR10] numérique pour les calculs d'interaction fluide-structure entre un écoulement laminaire incompressible et une structure élastique.

L'ensemble baigne dans un fluide en écoulement. La présence du cylindre fixe crée des recirculations qui mettent la lame en vibration. La géométrie utilisée avec les dimensions caractéristiques est présentée ci-dessous.

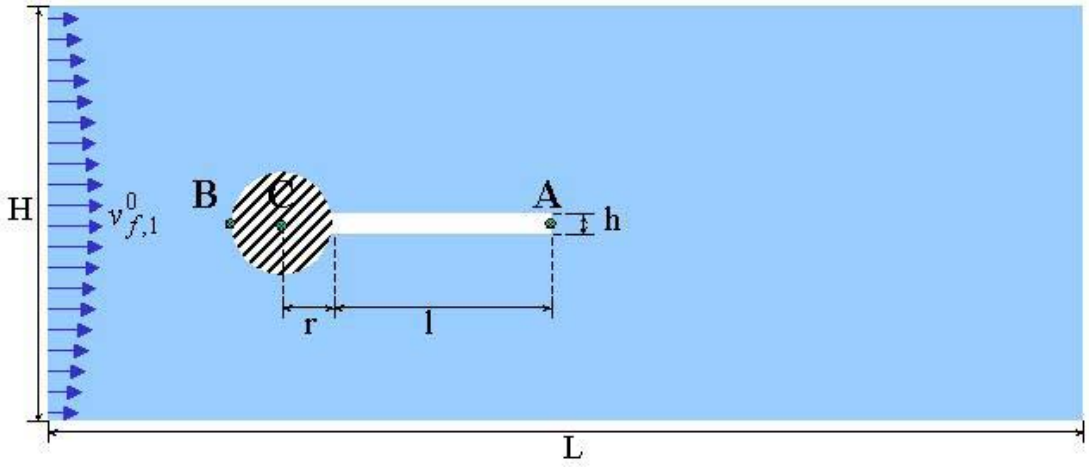

Figure 6 : Géométrie du cas

La hauteur du canal H est de 0.41 m. Le diamètre du cylindre vaut 0.1 m. La longueur de la lame flexible est de 0.35 m et son épaisseur de 0.02 m. La configuration correspondant à ces caractéristiques physiques est instable. Le but de la simulation est de tenter de reproduire le comportement de la lame tel que prédit dans la littérature (cas traité dans le cadre d'un benchmark pour lequel on dispose de données numériques et expérimentales).

4. Les codes de calcul du couplage

Le couplage fluide-structure est réalisé en utilisant le coupleur YACS développé dans le cadre du WP1.3 (livrable L2.3.2) et intégré dans la version de développement 4.0 de la plate-forme SALOME.
Ce couplage utilise plusieurs composants de calculs dont les deux principaux ont pour base le code de thermohydraulique *Code_Saturne* et le code de mécanique *Code_Aster* pour la simulation numérique des vibrations de structures sous écoulements. Les autres composants utilisés sont des modules qui réalisent les projections des champs de déplacements et de forces entre maillages incompatibles (modules ASTSAT et SATAST).

4.1 Code_Aster

Code_Aster est un code général pour l'étude du comportement mécanique des structures.
Le domaine d'application prioritaire est celui de la mécanique des solides déformables. Cependant, l'étude du comportement mécanique des composants industriels nécessite préalablement la modélisation des sollicitations auxquels ils sont soumis, ou des phénomènes physiques qui modifient les paramètres de ce comportement (fluide interne ou externe, température, changement de phases métallurgiques, efforts d'origine électro-magnétique...).
Pour ces raisons, le *Code_Aster* offre plusieurs possibilités de "chaînage" du phénomène mécanique avec les phénomènes thermique ou acoustique, ou avec des logiciels externes.
La version de base utilisée pour la réalisation du couplage IFS est *Code_Aster* version 9.0 disponible en Open Source sur le site http://www.code-aster.org/.

4.2 Code_Saturne

Code_Saturne est un code de mécanique des fluides numérique qui permet de résoudre les équations de Navier-Stokes pour des écoulements 2D ou 3D, stationnaires ou instationnaires, laminaires ou turbulents, incompressibles ou dilatables, isothermes ou non.
La discrétisation est de type volumes finis et permet l'utilisation d'une large gamme de maillages non structurés.
Code_Saturne permet notamment le traitement des maillages déformables par méthode ALE (Arbitrary Lagrangian Eulerian). Il peut ainsi modéliser l'effet de déformations de structures sur l'écoulement fluide et être couplé avec des codes de mécanique des structures comme *Code_Aster*.
La version de base de *Code_Saturne* utilisée dans le cadre de ce travail est la version 1.3.e.
Code_Saturne est disponible en open source sur le site http://rd.edf.com/code_saturne.

5. Le couplage

Le principe général du couplage est le suivant : *Code_Aster* et *Code_Saturne* s'exécutent séparément et s'échangent des informations à chaque pas de temps comme ci-dessous.

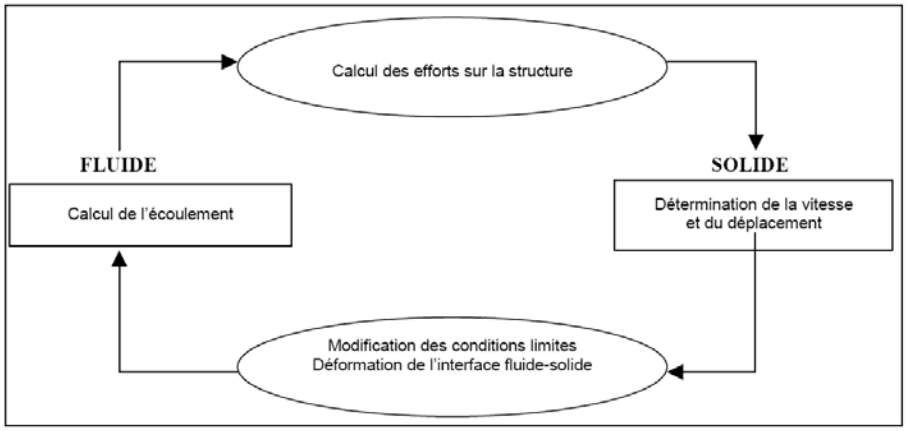

Figure 7 : Principe du couplage

Code_Saturne reçoit en entrée l'état de la structure (changement des conditions limites par rapport à l'itération précédente) et fournit l'écoulement fluide.
Code_Aster reçoit les chargements qui s'exercent sur la structure et fournit les déplacements et déformations qui en découlent.
Les maillages fluide et structure peuvent être discrétisés différemment (des convertisseurs assurant les interpolations nécessaires).

5.1 Échange de données

Comme les maillages de la structure et du fluide sont supposés incompatibles, on fait appel à des composants de projection des champs (ou convertisseurs) d'un maillage sur l'autre selon le principe suivant :

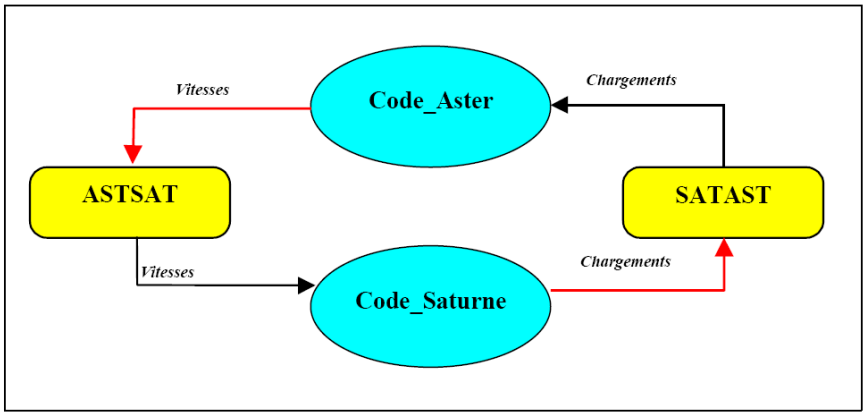

Figure 8 : Principe de l'échange de données

Code_Saturne fournit à *Code_Aster*, par l'intermédiaire d'un convertisseur (appelé SATAST), le chargement sur la structure considérée. *Code_Aster* fournit à *Code_Saturne*, par l'intermédiaire d'un convertisseur (appelé ASTSAT), la vitesse ou le déplacement de la

structure (voir figure 8, ci-dessus). Les méthodes utilisées ne sont pas décrites ici. On pourra consulter cette référence [DR12], par exemple.

5.2 Calcul transitoire et algorithme de couplage

Le problème du calcul statique couplé initial n'est pas traité ici. On ne traitera que le problème du calcul transitoire fluide-structure initialisé indépendamment pour la structure et le fluide.

Les algorithmes de couplage dans ce domaine sont nombreux car il s'agit d'un problème difficile. On peut rencontrer des instabilités numériques (divergence du calcul) ou des pertes de précision telles que l'énergie mécanique est fortement réduite en peu de temps. Il faut noter cependant que le choix de l'algorithme de couplage n'a pas d'influence sur la structure informatique du couplage entre les deux codes, mais concerne juste, au sein de chaque code, le choix (et éventuellement le type, le nombre et les instances) des données envoyées à l'autre code.

On citera quelques algorithmes possibles (voir références) :

- explicite synchrone [DR13], [DR14]
- explicite asynchrone [DR15]
- implicite [DR16]
- semi-implicite

C'est ce dernier qui a été retenu pour le démonstrateur. Il n'est pas parfaitement stable, comme on le verra sur les résultats, mais il donne des résultats satisfaisants dans de nombreux cas pour un effort de mise en œuvre réduit.

Son principe (déroulement temporel) est le suivant :

1. *Code_Saturne* envoie *à Code_Aster* le champ de force au temps tn e*t Code_Aster* s'en sert pour calculer le déplacement au temps tn+1
2. *Code_Aster* envoie à *Code_Saturne* le déplacement (et la vitesse de déplacement) des structures au temps tn+1. *Code_Saturne* s'en sert pour calculer le champ fluide au temps tn+1
3. Le processus est alors recommencé pour le pas de temps suivant.

Cet algorithme de couplage ne nécessite pas de sous-itérations mais il peut être instable. Aussi sera-t-il complété prochainement par un couplage permettant d'introduire des sous-itérations.

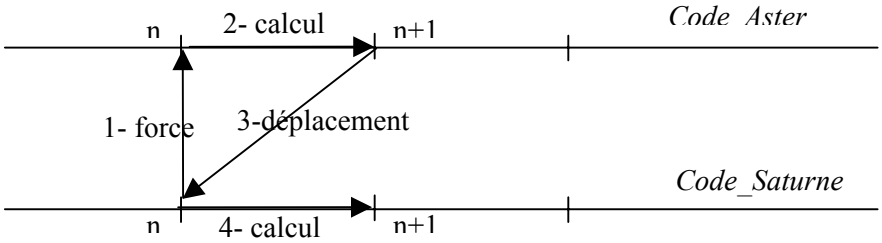

Figure 9 : Principe de l'échange temporel entre *Code_Saturne* et *Code_Aster*

6. Le couplage informatique

Le couplage fluide-structure est réalisé en utilisant le coupleur YACS développé dans le cadre du WP1.3 et intégré dans la version de développement 4.0 de la plate-forme SALOME.

Le point de départ est un couplage existant entre *Code_Aster* et *Code_Saturne* avec l'outil CALCIUM [DR3]. Le travail a consisté à adapter ce couplage à la nouvelle architecture fournie par le coupleur YACS.

On rappelle tout d'abord les outils de couplage disponibles dans YACS [DR11] puis leur mise en œuvre dans les composants de calcul.

6.1 Les outils de couplage de YACS

Dans le contexte de YACS et de la plate-forme SALOME, un couplage consiste en un assemblage de composants de calcul. Le superviseur est l'outil qui réalise l'assemblage des composants. Les composants de calcul doivent être construits en utilisant les services de base de la plate-forme SALOME.

On rappelle que la plate-forme SALOME utilise un intergiciel, à la norme CORBA, (omniORB) qui supporte les langages C++ et Python. Les services de base de la plate-forme sont spécifiés avec le langage IDL de CORBA et activés sous la forme d'objets distribués hébergés par des serveurs d'objets.

6.1.1 Les caractéristiques des composants

Les *composants* de simulation de la plate-forme SALOME sont soit des bibliothèques dynamiquement chargeables (so en Unix) pour les composants implémentés dans les langages C, C++ ou Fortran soit des modules pour ceux implémentés en langage Python.

Ces bibliothèques ou modules fournissent un point d'entrée normalisé appelable. Il retourne un objet CORBA qui est une instance du composant en question.

L'instanciation des composants est réalisée par des serveurs CORBA appelés *containers* qui répondent à une interface précise qui ne sera pas détaillée ici. Il suffit de savoir que si la bibliothèque disposant du point d'entrée normalisé et le composant étant déclaré auprès de la plate-forme, il peut être déployé par l'intermédiaire des containers.

Un composant met à disposition du superviseur plusieurs fonctionnalités qu'il peut assembler avec des fonctionnalités d'autres composants. Ces fonctionnalités correspondent en général à de grandes fonctions du composant : calculer un état stationnaire, calculer un pas de temps. Elles sont appelées *services*. Il est parfaitement possible d'avoir un seul service (Run, par exemple) pour un code existant que l'on ne veut pas restructurer.

Les services sont assemblés au moyen de *ports* de connexion. Il existe plusieurs types de ports.

- Le type le plus basique est le port d'entrée sortie « dataflow ».

Un port d'entrée dataflow reçoit une donnée d'un port de sortie auquel il est connecté et la délivre au service qui ne peut démarrer que lorsque tous ses ports d'entrée ont été correctement alimentés.

Un port de sortie dataflow recueille une donnée calculée par le service à la fin de son exécution et la transmet à tous les ports d'entrée connectés.

Un port dataflow est décrit par son nom, le type de donnée supporté et le sens (entrée ou sortie).

La connexion de ces ports permet de spécifier un transfert de donnée mais également une condition d'exécution d'un service.

Dans le schéma ci-dessous, le service 1 est connecté par un port de sortie à un port d'entrée du service 2. Il sera donc exécuté avant le deuxième service.

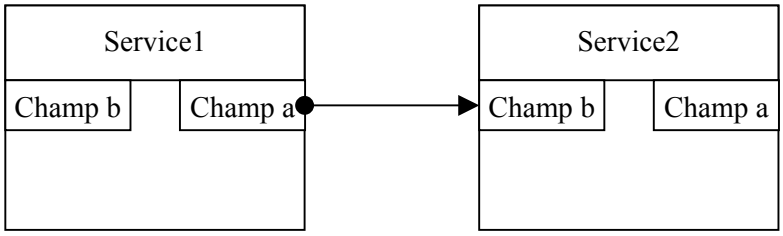

Figure 10 : Connexion par ports dataflow

- Le deuxième type de port de connexion pour un service est le port dit « datastream ».

Contrairement aux ports dataflow pour lesquels les transferts de données se situent à l'appel ou à la fin d'un service, les ports datastream permettent à un service de communiquer avec un autre service à tout moment de son exécution. Une utilisation typique de ce type de port est un service qui contient une boucle en temps. A chaque pas de temps, le service envoie des informations et attend des informations avant de pouvoir continuer à calculer.

Un port datastream est décrit par son nom, le type de donnée supporté, son sens (entrée ou sortie) et d'autres caractéristiques qui dépendent du type de port datastream utilisé. En effet, on peut avoir plusieurs types de port datastream avec des sémantiques différentes. Dans YACS, ne sont actuellement disponibles que les ports datastream de type CALCIUM (voir documentation CALCIUM [DR3] pour leur paramétrage).

Dans le schéma ci-dessous, le service 1 est connecté par des ports datastream au service 2. Le service 2 ne devra pas attendre la fin de service1 pour s'exécuter et ils échangeront des données au cours du calcul.

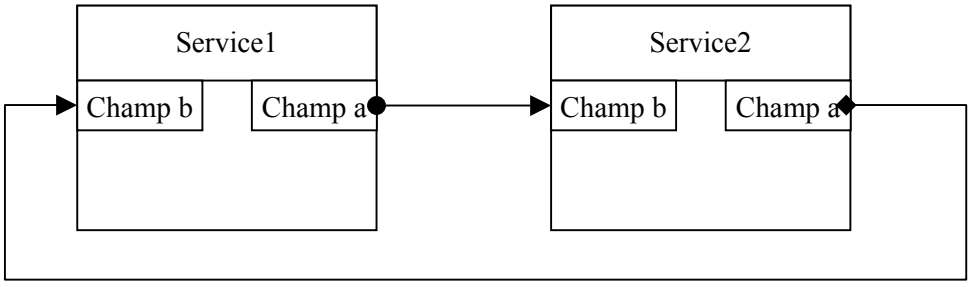

Figure 11 : Connexion par ports datastream

En général, on veut échanger des objets complexes entre codes couplés. Actuellement, les ports datastream de type CALCIUM ne permettent d'échanger que des tableaux de type simple (réel, entier, chaîne de caractères, booléen ou complexe). Il faut donc décomposer ces

objets complexes et les échanger au travers de plusieurs ports datastream. L'extension des ports datastream CALCIUM aux objets complexes (en particulier, maillages et champs) sera disponible à court terme.

6.1.2 L'assemblage des composants

L'assemblage des composants est réalisé par un outil appelé superviseur. Il lit un fichier de description du couplage (en langage XML), réalise les connexions décrites et déploie les composants sur les calculateurs disponibles en fonction des ressources disponibles.

On ne rentrera pas dans le détail de la syntaxe du fichier de configuration. Des exemples seront présentés plus tard lors de l'application au couplage *Code_Aster-Code_Saturne*.

6.2 Le composant Saturne

Le composant Saturne est doté de plusieurs ports datastream Calcium pour échanger les données suivantes :

- export du maillage fluide pariétal
- export du chargement sur ce maillage
- import du déplacement de la structure sur ce maillage
- import de la vitesse de la structure sur ce maillage
- import de l'indicateur de convergence des sous itérations

Les ports datastream CALCIUM pour réaliser ces échanges sont les suivants :

Nom du port	**Sens**	**Mode**	**Type**	**Commentaire**
dt	Out	Itération	Double	Pas de temps de calcul
dimension	Out	Itération	Entier	Dimensions du maillage pariétal
xmailsatur	Out	Itération	Double	Coordonnées des faces du maillage pariétal
xnodesatur	Out	Itération	Double	Coordonnées des nœuds du maillage pariétal
connect	Out	Itération	Entier	Connectivité du maillage pariétal
chargete	Out	Temps	Double	Forces sur les faces du maillage pariétal
viteface	In	Temps	Double	Vitesses des faces du maillage pariétal
deplnode	In	Temps	Double	Déplacement des nœuds du maillage pariétal
icv	In	Temps	Entier	Indicateur de convergence des sous itérations

Les échanges concernant le maillage pariétal (ports dimension, xmailsatur, xnodesatur, connect) et le pas de temps de calcul n'ont lieu qu'une fois au début du calcul (itération 1). Les autres échanges ont lieu à chaque pas de temps.

6.3 Le composant Aster

Le composant est doté de plusieurs ports datastream Calcium pour échanger les données suivantes :

- export du maillage pariétal de la structure
- export du déplacement de ce maillage
- export de la vitesse de ce maillage
- import des forces appliquées à la structure sur ce maillage
- import des moments appliqués à la structure sur ce maillage
- import de l'indicateur de convergence des sous itérations

Les ports datastream CALCIUM pour réaliser ces échanges sont les suivants :

Nom du port	Sens	Mode	Type	Commentaire
dimension	Out	Itération	Entier	Dimensions du maillage pariétal
noeuds	Out	Itération	Double	Coordonnées des nœuds du maillage pariétal
connect	Out	Itération	Entier	Connectivité du maillage pariétal
forcete	In	Temps	Double	Forces sur les nœuds du maillage pariétal
momentte	In	Temps	Double	Moments sur les nœuds du maillage pariétal
deforme	Out	Temps	Double	Déplacement des nœuds du maillage pariétal
vitesse	Out	Temps	Double	Vitesse des nœuds du maillage pariétal
icv	In	Temps	Entier	Indicateur de convergence des sous itérations

Les échanges concernant le maillage pariétal (ports dimension, noeuds, connect) n'ont lieu qu'une fois au début du calcul (itération 1). Les autres échanges ont lieu à chaque pas de temps.

6.4 Le composant SATAST

Ce composant est chargé de projeter les forces sur les faces du maillage pariétal du composant fluide en forces sur les nœuds du maillage pariétal de la structure. Il est doté de plusieurs ports datastream Calcium pour échanger les données suivantes :

- import du maillage pariétal de la structure
- import du maillage pariétal du fluide
- import des forces sur le maillage pariétal du fluide
- export des forces sur le maillage pariétal de la structure
- export des moments sur le maillage pariétal de la structure
- import de l'indicateur de convergence des sous itérations
- import du pas de temps de calcul fluide

Les ports datastream CALCIUM pour réaliser ces échanges sont les suivants :

Nom du port	Sens	Mode	Type	Commentaire
mailsat	In	Itération	Entier	Dimensions du maillage pariétal fluide
xmailsat	In	Itération	Double	Coordonnées des faces du maillage pariétal fluide
xnodesatur	In	Itération	Double	Coordonnées des nœuds du maillage pariétal fluide
iconsat	In	Itération	Entier	Connectivité du maillage pariétal fluide
mailast	Out	Itération	Entier	Dimensions du maillage pariétal structure
xmailast	Out	Itération	Double	Coordonnées des nœuds du maillage pariétal structure
iconast	Out	Itération	Entier	Connectivité du maillage pariétal structure
cha_in_te	In	Temps	Double	Forces sur les nœuds du maillage pariétal fluide
force_out_te	Out	Temps	Double	Force sur les nœuds du maillage pariétal structure
moment_out_te	Out	Temps	Double	Moment sur les nœuds du maillage pariétal structure
dt	In	Itération	Double	Pas de temps de calcul fluide
icv	In	Temps	Entier	Indicateur de convergence des sous itérations

Les échanges concernant les maillages (ports mailsat, xmailsat, xnodesatur, iconsat, mailast, xmailast, iconast) et le pas de temps de calcul n'ont lieu qu'une fois au début du calcul (itération 1). Les autres échanges ont lieu à chaque pas de temps.

6.5 Le composant ASTSAT

Ce composant est chargé de projeter les déplacements et vitesses connus sur les faces du maillage pariétal du composant structure en déplacements et vitesses sur les nœuds et faces du maillage pariétal du fluide. Il est doté de plusieurs ports datastream Calcium pour échanger les données suivantes :

- import du maillage pariétal de la structure
- import du maillage pariétal du fluide
- import des déplacements sur le maillage pariétal de la structure
- export des déplacements sur le maillage pariétal du fluide
- import des vitesses sur le maillage pariétal de la structure
- export des vitesses sur le maillage pariétal du fluide
- import de l'indicateur de convergence des sous itérations
- import du pas de temps de calcul fluide

Les ports datastream CALCIUM pour réaliser ces échanges sont les suivants :

Nom du port	Sens	Mode	Type	Commentaire
mailsat	In	Itération	Entier	Dimensions du maillage pariétal fluide
xmailsat	In	Itération	Double	Coordonnées des faces du maillage pariétal fluide
xnodesatur	In	Itération	Double	Coordonnées des nœuds du maillage pariétal fluide
iconsat	In	Itération	Entier	Connectivité du maillage pariétal fluide
mailast	Out	Itération	Entier	Dimensions du maillage pariétal structure
xmailast	Out	Itération	Double	Coordonnées des nœuds du maillage pariétal structure
iconast	Out	Itération	Entier	Connectivité du maillage pariétal structure
depla_in_te	In	Temps	Double	Déplacements des nœuds du maillage pariétal structure
nodedepla_out	Out	Temps	Double	Déplacements des nœuds du maillage pariétal fluide
vites_in	In	Temps	Double	Vitesse des nœuds du maillage pariétal structure
vites_out	Out	Temps	Double	Vitesse des faces du maillage pariétal fluide
dt	In	Temps	Double	Pas de temps de calcul fluide
icv	In	Temps	Entier	Indicateur de convergence des sous itérations

Les échanges concernant les maillages et le pas de temps de calcul n'ont lieu qu'une fois au début du calcul (itération 1). Les autres échanges ont lieu à chaque pas de temps.

6.6 Le composant Coupler

Ce composant est chargé de piloter le processus de sous itérations dans un pas de temps de calcul. Il reçoit les déplacements de la structure et les forces du fluide. Il calcule un indicateur de convergence et envoie cet indicateur dans des ports datastream CALCIUM qui peuvent être connectés aux autres composants pour piloter leur processus itératif interne. Ce composant n'intervient pas, pour le moment, dans ce couplage.

Ses ports datastream CALCIUM sont les suivants :

Nom du port	Sens	Mode	Type	Commentaire
mailsat	In	Itération	Entier	Dimensions du maillage pariétal fluide
mailast	In	Itération	Entier	Dimensions du maillage pariétal structure
depla_in_te	In	Temps	Double	Déplacements des nœuds du maillage pariétal structure
vites_in	In	Temps	Double	Vitesse des nœuds du maillage pariétal structure
cha_in_te	In	Temps	Double	Force sur les nœuds du maillage pariétal fluide
icv1	Out	Temps	Entier	Indicateur de convergence des sous itérations
icv2	Out	Temps	Entier	Indicateur de convergence des sous itérations

Les échanges concernant les maillages n'ont lieu qu'une fois au début du calcul (itération 1). Les autres échanges ont lieu à chaque pas de temps.

6.7 L'assemblage des composants pour le couplage

Les composants sont assemblés en connectant leurs ports datastream CALCIUM sortants avec un ou plusieurs ports de même type entrants. On liste ci-dessous les principales connexions réalisées.

Port source	Port cible	Commentaire
Aster.dimension	Astsat.mailast	
	Satast.mailast	
	Coupler.mailast	
Aster.noeuds	Astsat.xmailast	
	SatAst.xmailast	
Aster.connect	Astsat.iconast	
	Satast.iconast	
Aster.deforme	Astsat. depla_in_te	level=10
	Coupler. depla_in_te	level=10
Aster.vitesse	Astsat.vites_in	level=10
	Coupler.vites_in	level=10
Saturne.dimension	Astsat.mailsat	
	Satast.mailsat	
	Coupler.mailsat	
Saturne.xmailsatur	Astsat.xmailsat	
	Satast.xmailsat	
Saturne.xnodesatur	Astsat.xnodesatur	
	Satast.xnodesatur	
Saturne.connect	Astsat.iconsat	
	Satast.iconsat	
Saturne.dt	Astsat.dt	
	Satast.dt	
Saturne.chargete	Satast.cha_in.te	level=10
	Coupler.cha_in_te	level=10
Astsat.vites_out	Saturne.viteface	level=10
Astsat.nodedepla_out	Saturne.deplnode	level=10
Satast.force_out_te	Aster.forcete	level=10
Satast.Moment_out_te	Aster.momentte	level=10

Les ports datastream conservent par défaut toutes les données reçues lors des échanges pour être à même de faire des interpolations temporelles. Il est important de veiller à ne pas saturer la mémoire. Les ports susceptibles d'accumuler de grandes quantités de données au

cours du couplage ont été paramétrés pour limiter le nombre d'échantillons temporels (level=10) conservés.

La figure ci-dessous montre le schéma de couplage complet obtenu avec l'interface graphique YACS intégrée dans la plate-forme SALOME (livrable L1.3.2c développé par la société OpenCascade dans le cadre du WP1.3).

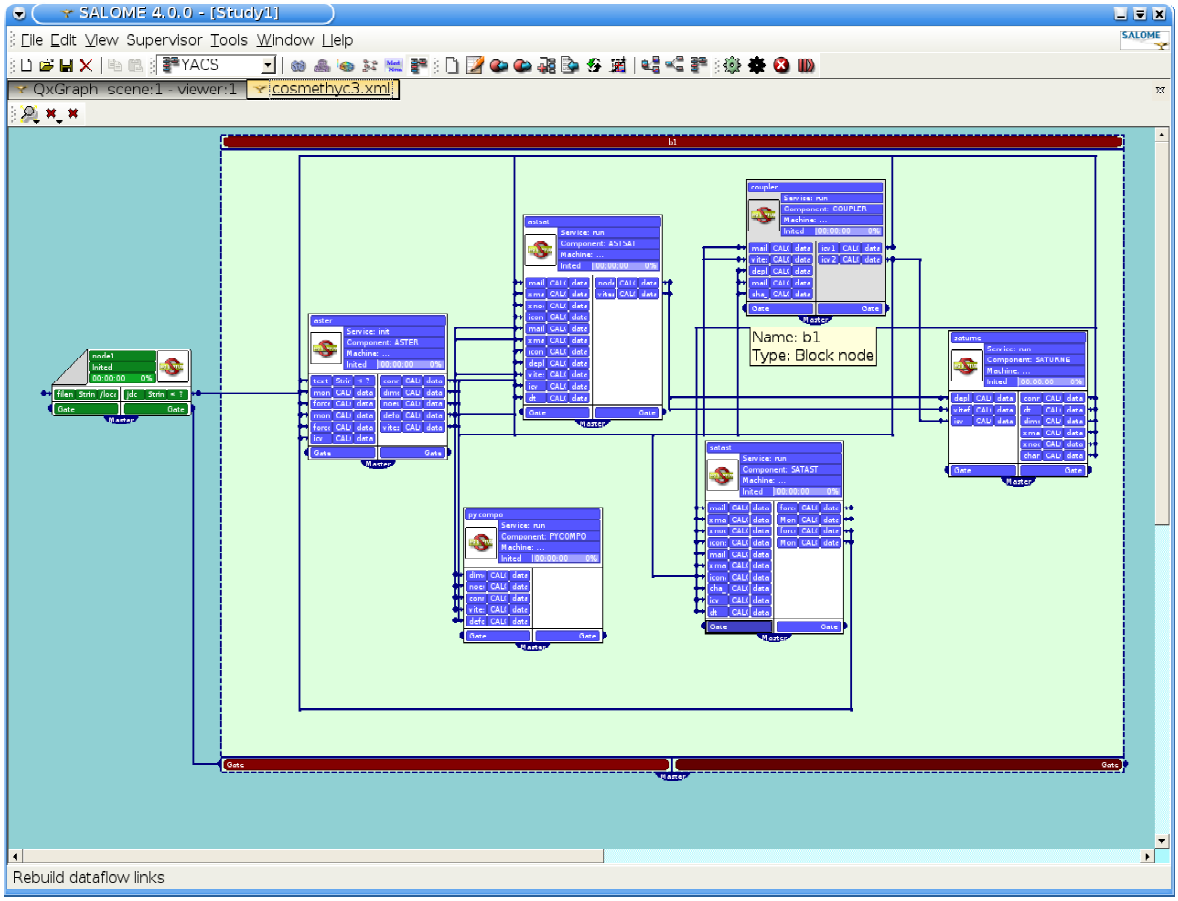

Figure 12 : Schéma de couplage dans l'IHM SALOME

7. Les résultats

7.1 Le faisceau

Le calcul de l'écoulement est réalisé sur un maillage 3D avec une seule maille en épaisseur (équivalent d'un calcul 2D). Le calcul de la structure est réalisé avec une poutre unidimensionnelle dans la direction z perpendiculaire au plan de l'écoulement.

Le maillage de l'écoulement de petite taille (3024 éléments hexaédriques) est présenté ci-dessous.

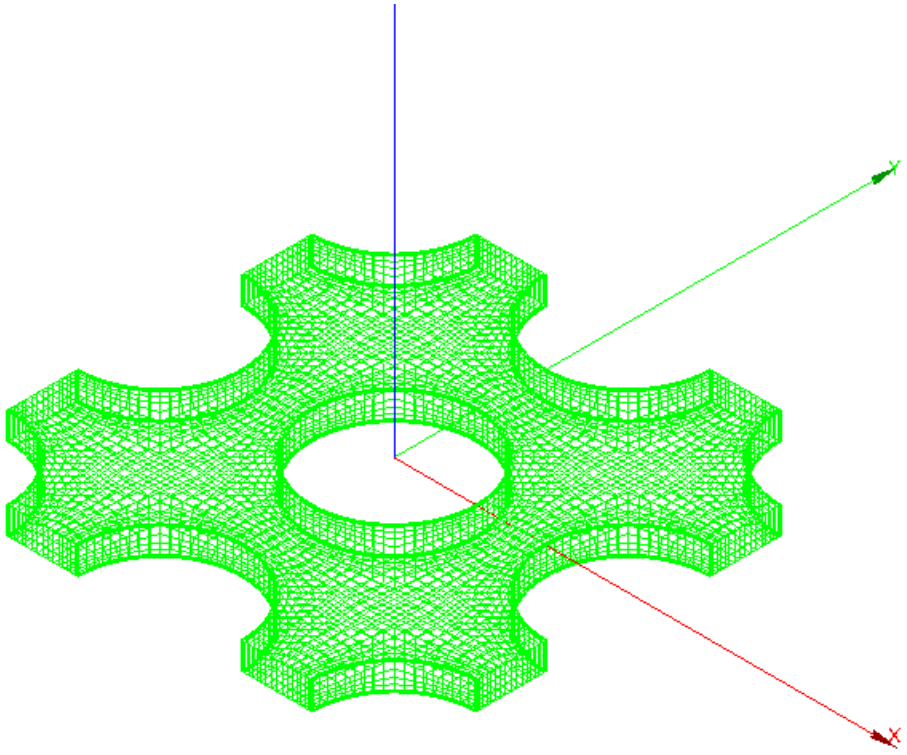

Figure 13 : Maillage du cas test faisceau

Le pas de temps retenu pour le calcul couplé transitoire est 0.001 s. Le transitoire est calculé pendant 300 pas de temps (0.3 s).

Les résultats obtenus sont identiques à ceux obtenus antérieurement avec le coupleur CALCIUM.

Voici quelques déformées du maillage à différents temps. Les déplacements ont été multipliés par 10. Les zones de déplacement maximal sont en rouge.

A 0.17 s, le tube est à son déplacement maximal dans la direction de l'écoulement (axe x).

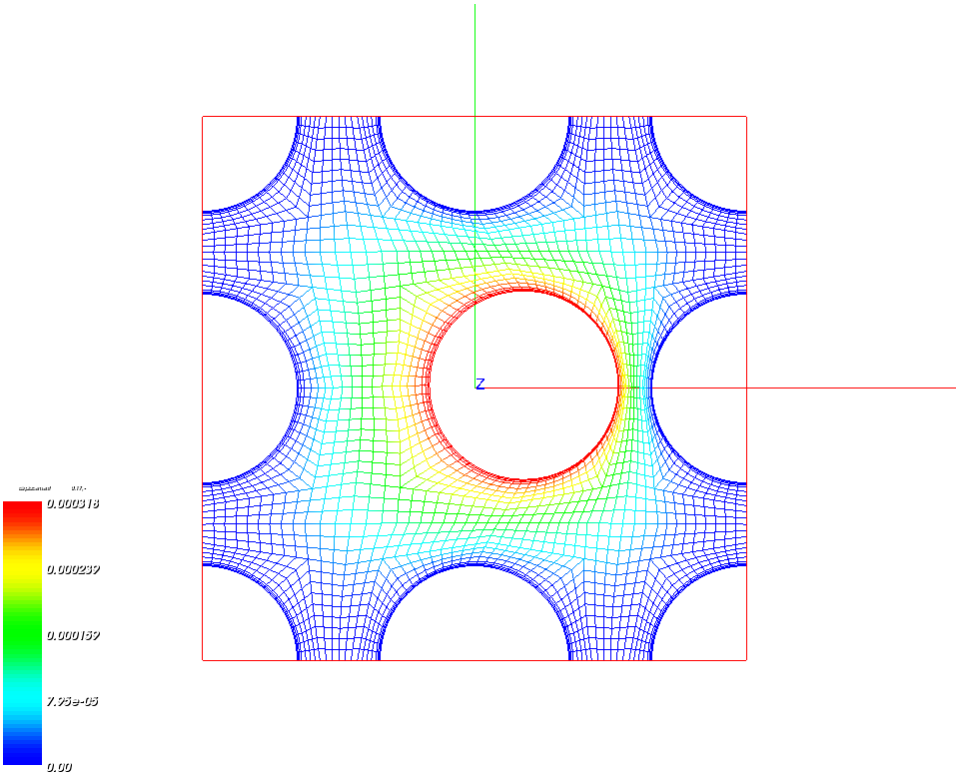

Figure 14 : Maillage déformé à 0.17 s

A 0.2 s, le tube est à son déplacement maximal dans le sens opposé à l'écoulement.

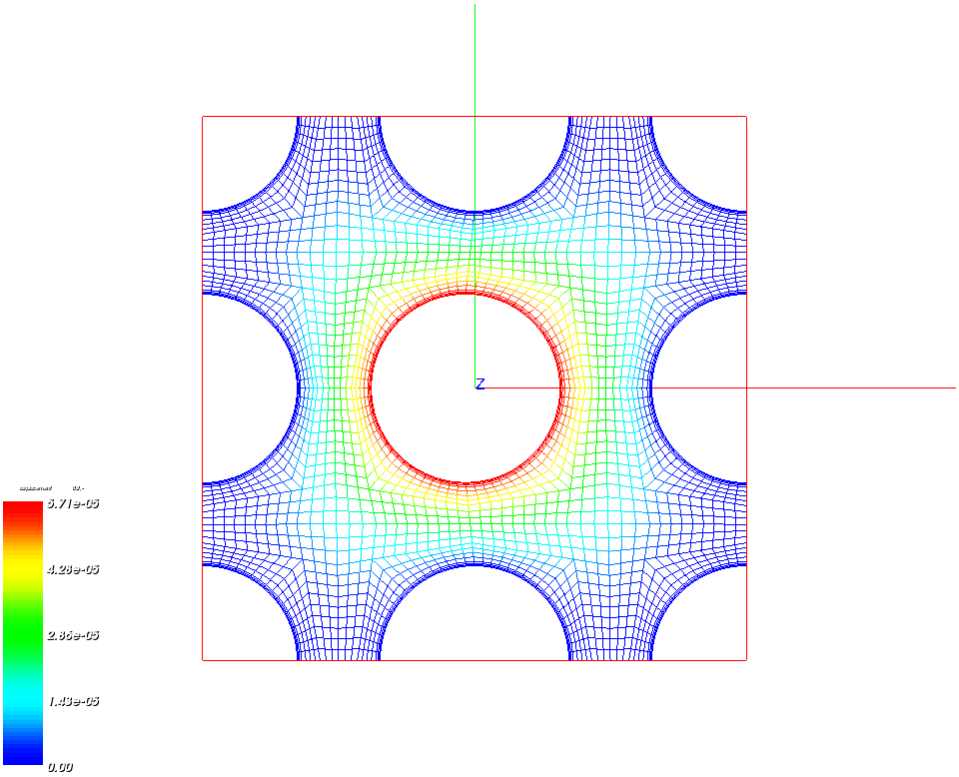

Figure 15 : Maillage déformé à 0.2 s

La vitesse maximale à 0.2 s est d'environ 8 m/s

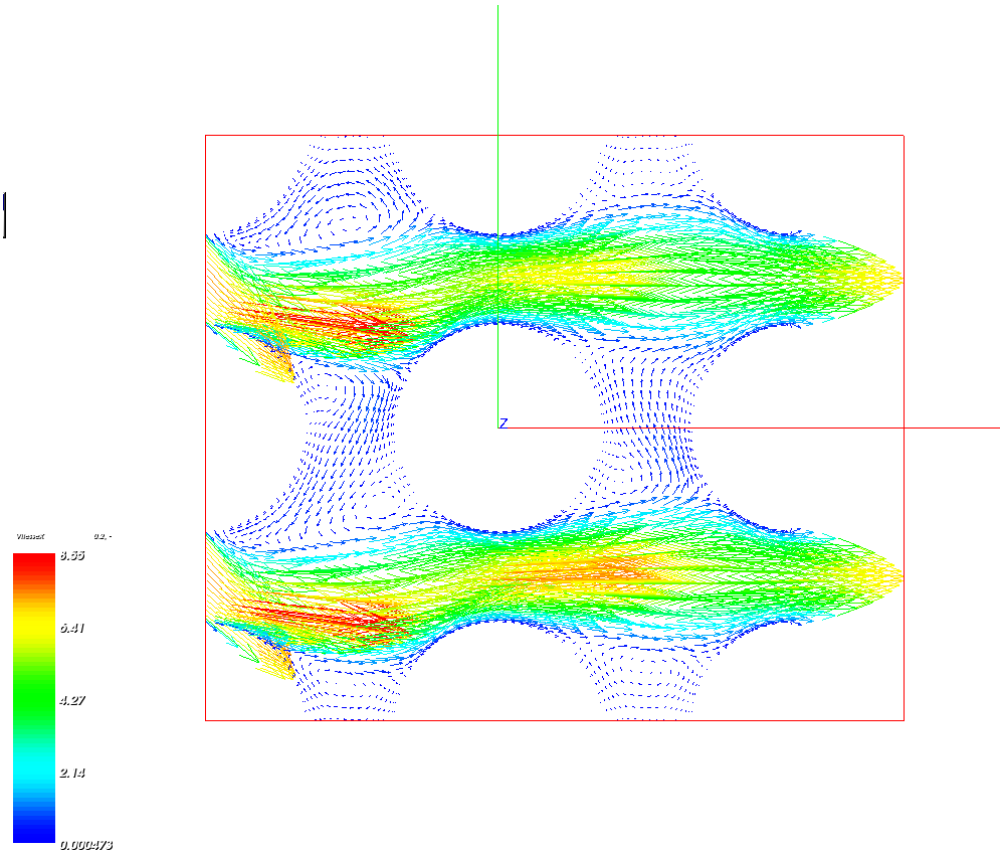

Figure 16 : Champ de vitesses à 0.2 s

7.2 Le tuyau

Le calcul de l'écoulement est réalisé sur un maillage 3D avec une seule maille en épaisseur (équivalent d'un calcul 2D). Le calcul de la structure est réalisé avec une poutre unidimensionnelle parallèle à l'écoulement.

Le maillage de l'écoulement est de petite taille (2142 éléments hexaédriques).

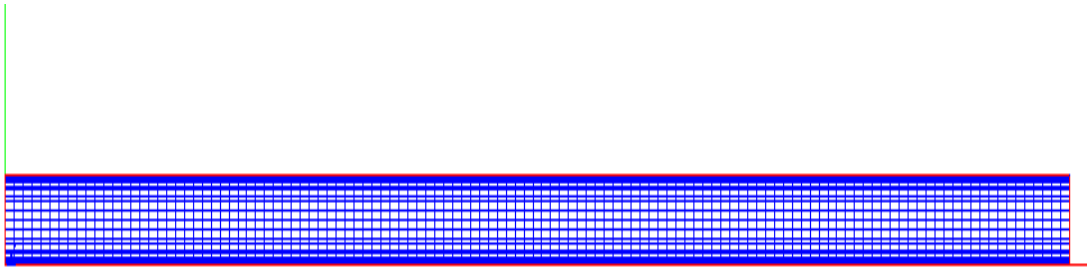

Figure 18 : Maillage du cas test tuyau flexible

Le pas de temps retenu pour le calcul couplé transitoire est 0.0001 s. Le transitoire est calculé pendant 53000 pas de temps (5.3 s de simulation) en démarrant avec un écoulement initial résultant d'un calcul non couplé. Le tuyau est déplacé au début de la simulation et on observe le déplacement du tuyau pour revenir à l'équilibre.

Les résultats obtenus sont identiques à ceux obtenus antérieurement avec le coupleur CALCIUM.

Voici quelques déformées du maillage à différents temps. Les déplacements ont été multipliés par 50000. Le champ de pression est représenté en même temps que la déformation du tuyau.

Au bout de 0.3 s de simulation, le tube est à un maximum de déformation.

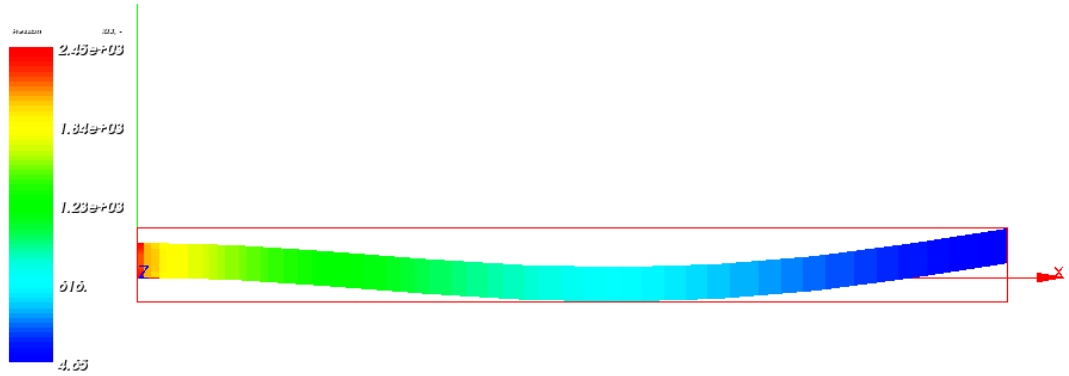

Figure 19 : Tuyau flexible déformé à 0.3 s

Au bout de 1.11 s de simulation, la déformation s'est inversée

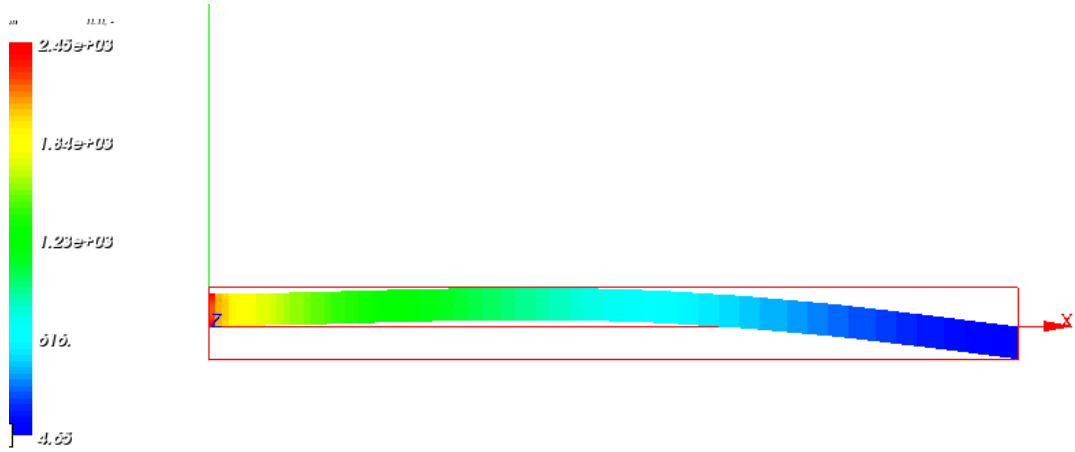

Figure 20 : Tuyau flexible déformé à 1.11 s

Puis elle s'amortit progressivement jusqu'à l'équilibre.

7.3 La lame flexible

Le calcul de l'écoulement est réalisé sur un maillage 3D avec une seule maille en épaisseur. Le calcul de la structure est réalisé avec un maillage 3D à base d'hexaèdres.

Le maillage de l'écoulement de taille moyenne (39116 éléments hexaédriques) est présenté ci-dessous (seulement la partie autour de la lame). Le maillage de structure comporte lui 400 éléments.

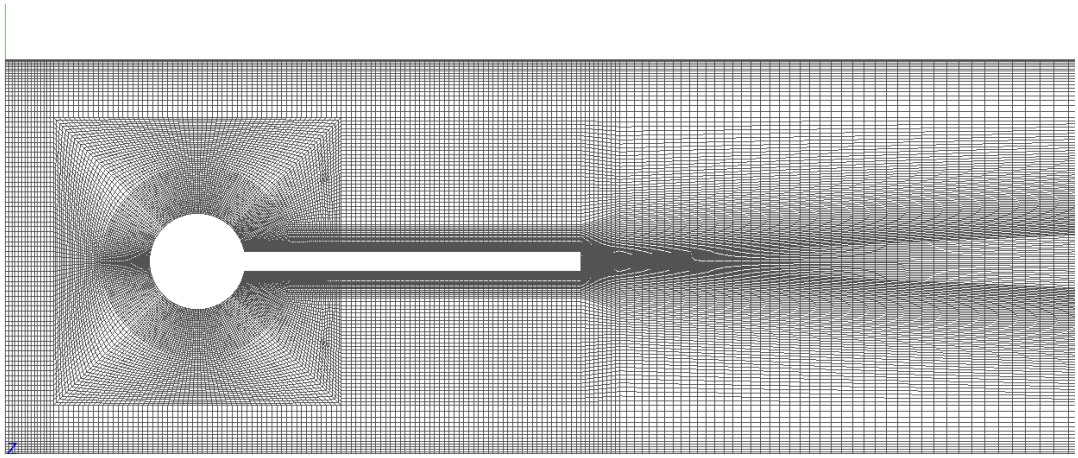

Figure 21 : Maillage du cas test lame flexible

Le pas de temps retenu pour le calcul couplé transitoire est 0.0005 s. Le transitoire est calculé pendant 90 pas de temps (0.045 s de simulation) en démarrant avec un écoulement initial résultant d'un calcul non couplé. Le calcul montre assez rapidement des instabilités ce qui explique l'arrêt au bout de seulement 90 pas de temps ce qui est trop rapide par rapport au départ en instabilité physiquement attendu. Ceci est une conséquence attendue de l'utilisation d'un schéma de couplage utilisé. Une méthode de couplage plus forte sera utilisée ultérieurement en faisant des sous itérations à chaque pas de temps.

On présente, ci-dessous, quelques déformées du maillage à différents temps. Les déplacements ont été multipliés par 2000. Le champ de pression est représenté en même temps que la déformation de la lame.

Au bout de 15 pas de temps (0.0075 s de simulation)

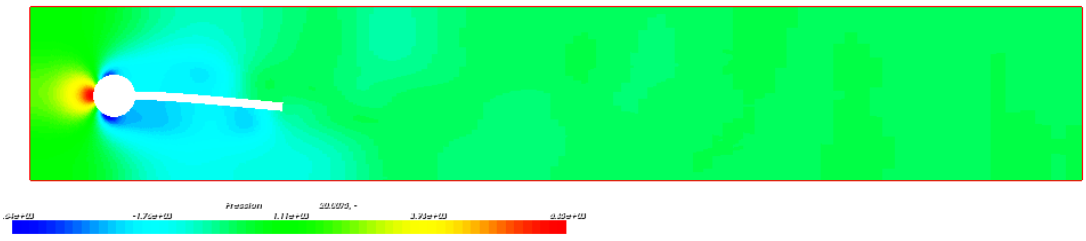

Figure 22 : Champ de pression et déformée à 0.0075 s

Après 20 pas de temps (0.01 s de simulation)

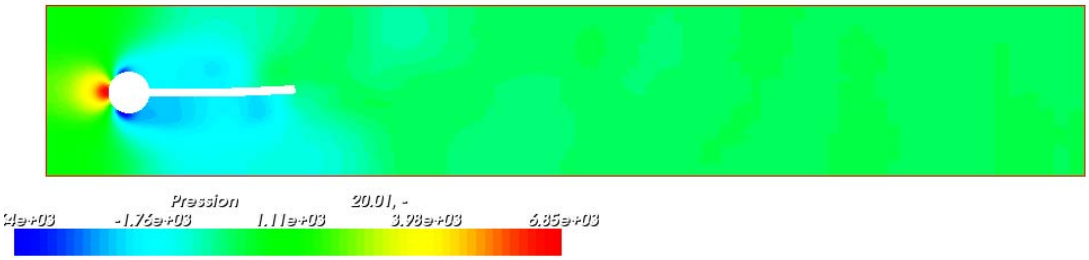

Figure 23 : Champ de pression et déformée à 0.01 s

Enfin, le champ de vitesse après 15 pas de temps

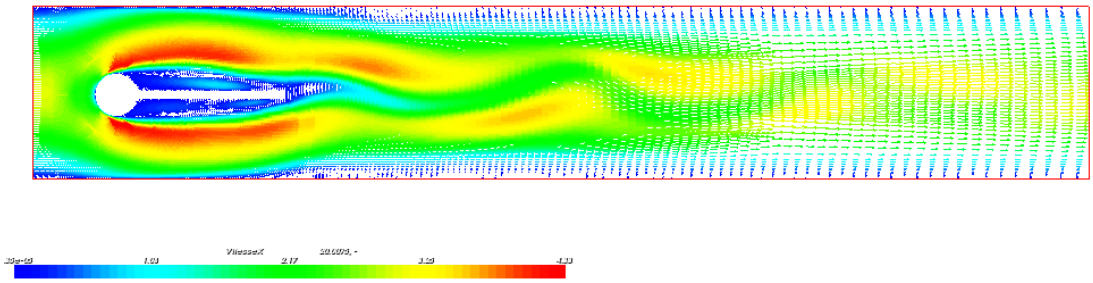

Figure 24 : Champ de vitesse à 0.0075 s

8. Réalisation informatique

Dans ce chapitre, on expose les aspects de la réalisation informatique liés à l'implémentation dans la plate-forme SALOME.

Pour réaliser le couplage décrit précédemment dans la plate-forme SALOME, il faut créer plusieurs composants SALOME qui seront regroupés dans un module SALOME. Ce module sera ajouté au lancement de la plate-forme SALOME à la liste des modules de base (GEOM, SMESH, VISU).

8.1 Le module COSMETHYC

Un module SALOME contient un ou plusieurs composants. La structure générale (répertoires, fichiers) du module COSMETHYC est la suivante :

```
COSMETHYC_SRC (répertoire racine)
      fichiers pour compilation, construction
      idl
            COSMETHYC.idl
      resources
            COSMETHYCCatalog.xml
            fichiers de couplage
      src
            ASTER
                  Fichiers du composant ASTER
            CONV
                  Fichiers des composants SATAST, ASTSAT, COUPLER
            SATURNE
                  Fichiers du composant SATURNE
            SATUYAU
                  Fichiers du composant SATUYAU
            SALAME
                  Fichiers du composant SALAME
            PYCOMPO
                  Fichiers du composant PYCOMPO
```

Le module contient en tout 8 composants SALOME : un composant pour Aster (répertoire ASTER), 3 composants pour Saturne (répertoire SATURNE, SATUYAU, SALAME), un composant pour le convertisseur SATAST (répertoire CONV), un composant pour le convertisseur ASTSAT (répertoire CONV), un composant pour COUPLER (répertoire CONV) et un composant de test (répertoire PYCOMPO).

On a donc un composant SALOME par composant du couplage sauf pour Saturne car pour chaque cas de calcul, on a des fichiers sources (fortran utilisateur) différents.

La procédure de configuration, compilation et installation est gérée en utilisant les outils autoconf et automake. Les sources sont gérés en configuration avec l'outil subversion.

8.2 Les composants SATAST, ASTSAT et COUPLER

Ces trois composants sont semblables et sont intégrés dans SALOME en suivant les mêmes principes. Ils sont regroupés dans un même répertoire car ils partagent un grand nombre de

fichiers sources. On ne détaillera qu'un seul des composants : SATAST. Les deux autres seront rapidement présentés.

SATAST, à l'origine, est un programme C. Il sera donc transformé en un composant SALOME C++ implémenté dans une bibliothèque dynamique. Cette bibliothèque dynamique doit jouer le rôle de fabrique de composant SALOME. Elle doit disposer d'un point d'entrée normalisé pour être utilisable par la plate-forme SALOME.

La convention est la suivante :

1. la bibliothèque a un nom normalisé : libSATASTEngine.so, pour le composant SATAST
2. il existe un point d'entrée de nom : SATASTEngine_factory qui a une signature imposée et qui retourne un objet CORBA qui sera le composant SATAST connu de la plate-forme. Il peut exister plusieurs exemplaires (instances) du composant SATAST. Chaque appel au point d'entrée (ou fabrique) en crée un nouveau.

La classe C++ qui représente le composant SATAST est définie dans les fichiers SATAST.hxx et SATAST.cxx.

8.2.1 La définition des services SALOME

Un composant SALOME a des services qui sont utilisés par la plate-forme SALOME pour l'exécuter. Le composant SATAST a un seul service de nom run sans aucun paramètre d'entrée et de sortie. On ne veut ici que reproduire le lancement d'un exécutable.

Ceci doit être décrit aussi bien au niveau CORBA qu'au niveau SALOME.

Au niveau CORBA, il faut décrire le composant dans le fichier COSMETHYC.idl contenu dans le répertoire idl). Sa description est la suivante :

```
interface SATAST: Engines::Superv_Component
{
  void run() raises (SALOME::SALOME_Exception);
};
```

Au niveau Salome, il faut décrire le composant dans le fichier COSMETHYCCatalog.xml. C'est une description équivalente au format XML qui a été faite ici à la main mais qui peut être produite automatiquement avec l'outil idlParser de SALOME.

Comme le composant SATAST a des ports datastream, la classe C++ qui le représente doit hériter de la classe de base SALOME : Superv_Component_i.

8.2.2 La création des ports datastream

Les ports datastream du composant SATAST doivent être créés dans une méthode de nom init_service de la classe C++ implémentation du composant SATAST. Chaque service du composant a ses propres ports. La création des ports datastream du service « run » du composant SATAST s'écrit comme suit :

```
CORBA::Boolean SATAST_i::init_service(const char * service_name){
CORBA::Boolean rtn = false;
string s_name(service_name);
  if (s_name == "run"){
    create_port(this,"mailsat","CALCIUM_integer","IN","I");
    create_port(this,"xnodesatur","CALCIUM_double","IN","I");
```

```
    create_port(this,"iconsat","CALCIUM_integer","IN","I");
    create_port(this,"mailast","CALCIUM_integer","IN","I");
    create_port(this,"xmailast","CALCIUM_double","IN","I");
    create_port(this,"cha_in_te","CALCIUM_double","IN","T");
    create_port(this,"force_out_te","CALCIUM_double","OUT","T");
    create_port(this,"icv","CALCIUM_integer","IN","T");
    create_port(this,"dt","CALCIUM_double","IN","I");
    rtn = true;
  }
return rtn;};
```

Toutes les créations de port n'apparaissent pas dans l'encadré ci-dessus car cette opération est assez répétitive. L'appel à create_port crée un port datastream pour le composant courant (this) en lui donnant un nom (mailsat ou autre) avec un type de donnée précis (CALCIUM_integer, CALCIUM_double ou autre), entrant ou sortant (IN ou OUT) et dépendant du temps ou d'une itération (T ou I).

8.2.3 L'utilisation des ports datastream

Le composant SATAST est implémenté en langage C. Il utilise l'interface d'appel C aux ports datastream. Ces appels sont regroupés dans le fichier Couplage.c. Le principe général d'utilisation est le suivant :

```
/* initialisation */
INFO=cp_cd(getCompo(),NOM_INSTANCE);

/*   lecture du pas de temps */
INFO=cp_ldb(getCompo(),CP_ITERATION,&Ti,&Tf,&ITER,"dt",IDIM,&NVAL,
&DTREF);

/* LECTURE DU NOMBRE DE MAILLES    */
INFO = cp_len (getCompo(), CP_SEQUENTIEL, &Tlue, &Tlue, &ITER,
"mailsat", IDIM, &NVAL, &Mail->NbNoeuds);

while(..le calcul continue..)
{
/* lecture des forces fluides */
INFO=cp_ldb(getCompo(),CP_SEQUENTIEL,&Ti,&Tf,&ITER,"cha_in_te",
IDIM, &NVAL, Vect);
if (INFO != CPOK)cp_exit(CPATAL);/* sortie par deconnexion */

... projection des forces fluides sur le maillage structure...

/* ecriture des forces sur la structure .. */
INFO=cp_edb(getCompo(),CP_TEMPS,Ti,ITER,"force_out_te",IDIM, Force);
}
```

Quasiment toutes les fonctions de l'interface C prennent en premier argument un pointeur sur le composant qui porte les ports datastream. Un composant implémenté en C (ou en Fortran d'ailleurs) ne dispose pas de cette information de manière intrinsèque comme en C++. Il faut donc une mécanique spécifique pour retrouver ce pointeur. La solution retenue, ici, est d'ajouter à chaque bibliothèque dynamique deux points d'entrée qui permettent :

1. de mémoriser le pointeur sur le composant : setCompo(pointeur)
2. de retrouver le pointeur mémorisé : getCompo()

Cette mécanique a ses limites : il est possible de créer des composants de types différents (SATAST et ASTSAT, par exemple) dans un même processus mais il n'est pas possible d'en créer plusieurs du même type (2 composants de type SATAST, par exemple).

Ensuite, lecture et écriture de données sur un port se font comme avec l'outil CALCIUM existant.

La lecture (cp_lXX) peut être temporelle, itérative ou séquentielle (CP_TEMPS, CP_ITERATION, CP_SEQUENTIEL). L'utilisation des arguments suivants (ti, tf, iter) dépend du cas. Il faut évidemment donner le nom du port. Enfin les derniers arguments permettent de récupérer dans un tableau les valeurs lues. Le type des données lues (XX=en, re, db, ...) doit être celui déclaré à la création du port.

L'écriture (cp_eXX) est plus simple. Elle ne peut être que temporelle ou itérative avec seulement 2 arguments (t, iter), le nom du port et le tableau écrit avec sa taille.

8.2.4 Traitement des erreurs et sortie

Les erreurs et la sortie doivent être gérées avec précaution. En effet, dans un programme C, lorsqu'on rencontre une erreur, il est fréquent de terminer brutalement le programme par un exit(1) ou quelque chose de semblable. Un composant SALOME ne doit pas se comporter de cette façon. Il doit de préférence lever une exception C++ qui pourra être récupérée par SALOME, interprétée comme une erreur et surtout n'arrêtera pas la plate-forme.

L'appel cp_exit(code_erreur) fourni par SALOME permet de lever une exception C++ à partir du C. Il est donc recommandé d'appeler cette fonction au lieu de faire un appel à exit(). Ceci ne règle cependant pas complètement le problème. En effet, en général, les exceptions C++ ne sont pas propagées, par défaut, au travers du C ou du Fortran. Plusieurs compilateurs dont les compilateurs GNU et Intel offrent cette possibilité (option du compilateur –fexceptions). Cette option a été utilisée pour tous les composants en C et en Fortran.

8.2.5 L'implémentation du service run

Avec tous les éléments précédents, il est maintenant possible d'implémenter le service run du composant SATAST.

```
void SATAST_i::run()
{
  setCompo(this);
  try{
      satast();
      cp_fin(this,CP_ARRET);
    }
  catch ( const CalciumException & ex){
      std::cerr << ex.what() << std::endl;
      cp_fin(this,CP_ARRET);
    }
  catch (...){
      std::cerr << "unexpected exception" << std::endl;
      cp_fin(this,CP_ARRET);
    }
}
```

On commence par mémoriser le pointeur sur le composant (appel de setCompo) puis on appelle l'implémentation C de SATAST dans un bloc de gestion d'exceptions C++. Le programme principal a été renommé de main en satast.

8.2.6 La déclaration du composant dans le fichier de couplage

Le composant doit également être déclaré dans le fichier de couplage (en langage XML) qui est lu par le coupleur YACS pour configurer le couplage. Cette déclaration sera, par la suite, automatiquement extraite du fichier COSMETHYCCatalog.xml.

Cette déclaration a la forme suivante :

```
<service name="satast" >
  <component>SATAST</component>
  <method>run</method>
  <load container="E"/>
  <instream name="mailsat" type="CALCIUM_integer"/>
  <instream name="xmailsat" type="CALCIUM_double"/>
  <instream name="xnodesatur" type="CALCIUM_double"/>
  <instream name="iconsat" type="CALCIUM_integer"/>
  <instream name="mailast" type="CALCIUM_integer"/>
  <instream name="xmailast" type="CALCIUM_double"/>
  <instream name="iconast" type="CALCIUM_integer"/>
  <instream name="cha_in_te" type="CALCIUM_double"/>
  <outstream name="force_out_te" type="CALCIUM_double"/>
  <outstream name="Moment_out_te" type="CALCIUM_double"/>
  <instream name="icv" type="CALCIUM_integer"/>
  <instream name="dt" type="CALCIUM_double"/>
</service>
```

On y dit que l'on veut créer un exemplaire du composant SATAST, exécuter son service run sur le container SALOME "E" et que ce service a plusieurs ports datastream entrants et sortants. Cette description permet au superviseur de vérifier avant lancement la validité des connexions entre ports.

8.2.7 Les composants ASTSAT et COUPLER

Les composants ASTSAT et COUPLER sont très semblables au composant SATAST : implémentation en C, lecture séquentielle, ... Il suffit de changer les noms des composants et des ports pour obtenir ces autres composants.

Comme on le constate rapidement, la fabrication d'un composant SALOME doté de ports datastream est assez systématique et pourrait donc être automatisée. Il existe un outil (HXX2SALOME) qui automatise la construction de composant SALOME sans port datastream à partir d'une classe C++. Une extension aux composants dotés de ports datastream est possible.

8.3 Les composants SATURNE

Le module COSMETHYC contient 3 composants SATURNE : un composant par cas de calcul (faisceau, tuyau, lame) car les conditions aux limites sont spécifiés dans des Fortrans utilisateurs qui sont donc différents selon les cas.

Il suffit d'analyser un seul des trois composants. Les deux autres sont très semblables.

Le premier composant de nom SATURNE est semblable au composant SATAST. On passera donc rapidement sur sa description.

8.3.1 Le composant

Il s'agit d'un composant SALOME C++ qui est donc implémenté dans une bibliothèque dynamique de nom libSATURNEEngine.so avec un point d'entrée de nom SATURNEEngine_factory.

8.3.2 Les services

Les composants SATURNE ont un service run qui correspond à l'exécution du programme et dont l'interface IDL est la suivante :

```
interface SATURNE: Engines::Superv_Component
{
  void run() raises (SALOME::SALOME_Exception);
};
```

8.3.3 Les ports datastream

La création des ports datastream du service « run » du composant SATURNE s'écrit comme suit :

```
CORBA::Boolean SATURNE_i::init_service(const char * service_name){
CORBA::Boolean rtn = false;
string s_name(service_name);
  if (s_name == "run"){
      create_port(this,"connect","CALCIUM_integer","OUT","I");
      create_port(this,"dt","CALCIUM_double","OUT","I");
      create_port(this,"dimension","CALCIUM_integer","OUT","I");
      create_port(this,"xmailsatur","CALCIUM_double","OUT","I");
      create_port(this,"xnodesatur","CALCIUM_double","OUT","I");
      create_port(this,"chargete","CALCIUM_double","OUT","T");
      create_port(this,"deplnode","CALCIUM_double","IN","T");
      create_port(this,"viteface","CALCIUM_double","IN","T");
      create_port(this,"icv","CALCIUM_integer","IN","T");
    rtn = true;
  }
return rtn;};
```

8.3.4 L'implémentation

Le composant SATURNE est implémenté en langages C et Fortran. Il utilise l'interface d'appel C aux ports datastream. Ces appels sont regroupés dans le fichier cs_astr_coupling.c.

Le programme principal main a été renommé en saturne. Les paramètres de la ligne de commande (longia, longra, param) sont spécifiés en dur dans le composant.

8.3.5 Déclaration dans le fichier de couplage

L'exécution du service run du composant SATURNE est décrite comme suit dans le fichier de couplage XML.

```
      <service name="saturne" >
        <component>SATURNE</component>
        <method>run</method>
```

```
        <load container="B"/>
        <outstream name="connect" type="CALCIUM_integer"/>
        <outstream name="dt" type="CALCIUM_double"/>
        <outstream name="dimension" type="CALCIUM_integer"/>
        <outstream name="xmailsatur" type="CALCIUM_double"/>
        <outstream name="xnodesatur" type="CALCIUM_double"/>
        <outstream name="chargete" type="CALCIUM_double"/>
        <instream name="deplnode" type="CALCIUM_double"/>
        <instream name="viteface" type="CALCIUM_double"/>
        <instream name="icv" type="CALCIUM_integer"/>
      </service>
```

8.4 Le composant ASTER

On a un seul composant pour les 3 cas de calcul. Chaque cas est paramétré par le fichier de commandes et les fichiers de données.

8.4.1 Le composant

Aster est un code à langage de commandes. Le langage de commandes est basé sur le langage interprété Python. La bibliothèque de calcul est codée principalement en Fortran. Elle est interfacée avec le langage Python au moyen d'un module Python (de nom aster) codé en C qui est lui même interfacé avec la bibliothèque Fortran. Ce module est habituellement linké statiquement avec l'exécutable Aster. Dans le contexte SALOME, ce module doit être construit sous la forme d'une bibliothèque dynamiquement chargeable (so).

Pour intégrer Aster en tant que composant SALOME couplable par ports datastream, on utilise par conséquent un composant SALOME Python. Sa logique d'intégration dans SALOME est très semblable à celle d'un composant C++, elle diffère seulement dans la forme.

Un composant SALOME Python est une classe Python dont le nom est celui du composant que l'on trouve dans un module Python dont le nom est également celui du composant (donc ASTER.py, ici). Il doit hériter de la classe PyDSCComponent contenue dans le module dsccalcium de SALOME.

8.4.2 Les services du composant

Le composant ASTER a deux services : init et run. Le service init qui est le seul utilisé ici sert lors d'une première exécution du fichier de commandes dont le contenu est passé en argument du service (text). Le service run permet de poursuivre l'exécution commencée avec init. Il reçoit également un texte qui contient des commandes Aster supplémentaires.

L'interface IDL est la suivante :

```
interface ASTER: Engines::Superv_Component
{
  void init(in string text) raises (SALOME::SALOME_Exception);
  void run(in string text) raises (SALOME::SALOME_Exception);
};
```

8.4.3 Les ports datastream

La création des ports datastream du service « init » s'écrit comme suit :

```
from calcium import create_port

def init_service(self,service):
  if service == "init":
    create_port(self.proxy,"connect","CALCIUM_integer","OUT","I")
    create_port(self.proxy,"dimension","CALCIUM_integer","OUT","I")
    create_port(self.proxy,"noeuds","CALCIUM_double","OUT","I")
    create_port(self.proxy,"deforme","CALCIUM_double","OUT","T")
    create_port(self.proxy,"vitesse","CALCIUM_double","OUT","T")
    create_port(self.proxy,"momentit","CALCIUM_double","IN","I")
    create_port(self.proxy,"forceit","CALCIUM_double","IN","I")
    create_port(self.proxy,"momentte","CALCIUM_double","IN","T")
    create_port(self.proxy,"forcete","CALCIUM_double","IN","T")
    create_port(self.proxy,"icv","CALCIUM_integer","IN","T")
    return True
```

8.4.4 L'implémentation du composant

Le composant ASTER utilise l'interface d'appel Fortran aux ports datastream. Le pointeur sur le composant est mémorisé dans le COMMON COUP1 (fichier couplage.h) au moyen de la subroutine setcompo.

La subroutine setcompo est appelée depuis le langage Python. Il a donc fallu ajouter une fonction de nom setcompo au module aster (fichier astermodule.c).

La réalisation du service init est plus complexe que pour les composants précédents car il a fallu intégrer une partie de l'interpréteur de commandes d'Aster. Il n'est pas sûr que cette façon de faire soit la meilleure mais elle fonctionne sans problème.

8.4.5 Traitement des erreurs et sortie

Contrairement aux composants décrits précédemment, les erreurs et la sortie de service ne sont pas gérés par levée d'une exception C++ mais en appelant la subroutine d'Aster UEXCEP qui lève une exception Python qui peut ensuite être récupérée dans SALOME.

8.4.6 Déclaration dans le fichier de couplage

L'exécution du service init du composant ASTER est décrite comme suit dans le fichier de couplage XML.

```
      <service name="aster" >
        <component>ASTER</component>
        <method>init</method>
        <load container="A"/>
        <inport name="text" type="string"/>
        <outstream name="connect" type="CALCIUM_integer"/>
        <outstream name="dimension" type="CALCIUM_integer"/>
```

```
        <outstream name="noeuds" type="CALCIUM_double"/>
        <outstream name="deforme" type="CALCIUM_double"/>
        <outstream name="vitesse" type="CALCIUM_double"/>
        <instream name="momentte" type="CALCIUM_double"/>
        <instream name="forcete" type="CALCIUM_double"/>
        <instream name="icv" type="CALCIUM_integer"/>
      </service>
```

8.5 Les couplages

Chaque couplage est décrit par un fichier XML (faisceau.xml, tuyau.xml, lame.xml dans le répertoire resources) qui contient les descriptions de services que l'on a vu précédemment et les connexions entre ports.

On présente ci-dessous deux exemples de connexion

```
    <stream>
      <fromnode>satast</fromnode> <fromport>force_out_te</fromport>
      <tonode>aster</tonode> <toport>forcete</toport>
      <property name="level" value="10"/>
    </stream>
    <stream>
      <fromnode>aster</fromnode> <fromport>deforme</fromport>
      <tonode>astsat</tonode> <toport>depla_in_te</toport>
      <property name="level" value="10"/>
    </stream>
```

Il faut indiquer quels nœuds (au sens nœud de calcul dans le fichier de couplage) et quels ports sont source et cible du lien de connexion. On peut paramétrer le lien avec des propriétés. On a spécifié ici que le lien doit conserver au plus 10 des items échangés (propriété level).

Tous les calculs ont été réalisés en plaçant chacun des composants dans un container (donc processus) différent sur une même machine. D'autres configurations sont possibles mais ont été peu utilisées.

9. Conclusion

Le nouveau modèle de couplage intégré dans le coupleur YACS et plus particulièrement le couplage par ports "datastream" a été utilisé avec succès pour coupler les codes *Code_Aster* et *Code_Saturne* en vue de modéliser des phénomènes d'interaction fluide-structure.

Trois cas de calcul ont été réalisés sur des géométries différentes : écoulement transverse à un faisceau de tubes, écoulement dans un tuyau souple et écoulement autour d'une lame flexible.

Ces calculs ont donnés des résultats identiques à de précédents calculs pour les deux premiers. Le troisième n'a pas pu être mené jusqu'au bout pour des raisons d'instabilités numériques. Le schéma de couplage retenu n'était pas suffisamment fort pour le cas simulé. Les résultats n'ont donc pas pu être comparés avec ceux de la littérature.

L'utilisation des ports "datastream" a permis de réaliser le couplage sans restructurer en profondeur les codes existants *Code_Aster* et *Code_Saturne*. L'intégration dans la plate-forme SALOME a été réalisée sans difficulté majeure. Elle demande quelques modifications mineures des codes et de prêter attention au traitement des erreurs. Huit composants de natures différentes ont été intégrés : des codes en C, des codes mixtes C et Fortran et enfin un code en Fortran interfacé avec le langage Python.

L'objectif initial est atteint mais pour aller au bout de la démonstration concernant l'apport du nouveau modèle de couplage, il reste encore quelques actions à mener :

- Rajouter au présent prototype un couplage plus fort entre les codes, par méthode de point fixe. Ainsi, des cas tels que celui de la lame flexible pourront arriver à convergence.
- Utiliser des données d'échange de plus haut niveau comme le modèle MED.
- Faire fonctionner le coupleur sur architecture parallèle car les calculs fluide sont très consommateurs en temps CPU. La structure de *Code_Saturne* est déjà prête. Quand il fonctionne en parallèle sur plusieurs processeurs, les données de couplage sont centralisées sur le processeur maître avant d'être envoyées au coupleur.
- Étudier l'apport du couplage d'interface dans la construction de couplages, en général, et pour l'interaction fluide-structure plus spécifiquement (plus haut niveau d'abstraction, en particulier).