



**Compte rendu de la formation YACS**  
**dispensée à EDF-Clamart du 21 au 23 septembre 2009 -**  
**Analogies avec PALM**

Référence : WN-CMGC-09-97

Auteurs : Anthony Thevenin, Sophie Ricci

Date : 25 Octobre 2009

Résumé :

Ce document a pour but de faire un compte rendu de la formation dispensée par EDF sur l'intégration et le couplage de codes de calcul dans SALOME par le biais du coupleur de codes YACS (module de la plateforme SALOME). Des analogies entre YACS et PALM seront discutées.

---



# SOMMAIRE

|   |    |
|---|----|
| I) INTRODUCTION.....  | 4  |
| II) PRESENTATION DE SALOME .....                                | 4  |
| III) SURVOL DE LA PLATEFORME VIA L'ETUDE D'UN CAS D'ECOLE ..... | 5  |
| IV) PRISE EN MAIN DE YACS .....                                 | 7  |
| V) UTILISATION DE YACSGEN .....                                 | 9  |
| VI) UTILISATION DE HXX2SALOME .....                             | 12 |
| VI) INTRODUCTION AU FORMAT MED .....                            | 13 |

## **I) INTRODUCTION**

Cette formation sur trois jours comportait le programme suivant :

- ↪ première demi-journée : présentation générale de SALOME puis traitement complet d'un cas d'école sous forme d'un TD,
- ↪ deuxième demi-journée : présentation générale du module de supervision YACS puis TD sur une première élaboration d'un schéma de calcul dans YACS,
- ↪ deuxième journée : présentation et utilisation de deux outils d'intégration de code dans SALOME : YACSGEN et HXX2SALOME,
- ↪ troisième journée : présentations et utilisations de MEDMEM (module d'échange de données) et des outils d'interpolation.

## **II) PRESENTATION DE SALOME**

SALOME est une plateforme de simulation numérique co-développée par EDF, le CEA et Open Cascade depuis 2001. Elle est majoritairement utilisée en interne par EDF et le CEA pour leurs besoins propres mais est disponible sous les termes de la licence GNU L-PGL. La politique de diffusion de SALOME entend répondre aux sollicitations extérieures sans pour autant viser une distribution intensive du code.

SALOME est composée de différents modules offrant à l'utilisateur des outils de pré et post traitement ainsi que la possibilité de faire de la supervision et du couplage de codes de calcul :

- ↪ GEOM : module utilisé pour faire de la CAO,
- ↪ SMESH : module utilisé pour la génération de maillage (permet de mailler une géométrie créée via le module GEOM),
- ↪ YACS : module utilisé pour faire de la supervision de schémas de calcul et pour le couplage
- ↪ VISU : module utilisé pour visualiser les résultats.

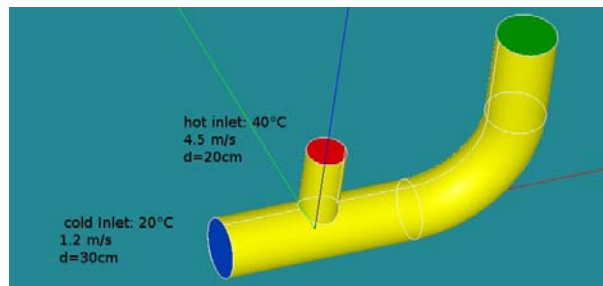
L'ensemble des échanges entre les modules de SALOME s'appuie sur le modèle MED. Ces échanges peuvent se faire soit par fichier au format MED qui inclut un formatage de l'information à la norme binaire HDF5 soit directement en mémoire via MEDMEM. Ce modèle permet d'échanger des données qui correspondent à des géométries, des maillages et des champs de résultats.

D'un point de vue technique, le modèle d'échange d'information en parallèle de SALOME est basé sur la technologie CORBA. L'architecture de SALOME s'appuie sur le concept de composants. Les composants (qui peuvent être écrits dans des langages de programmation distincts) sont assemblés afin de construire des applications complètes. Ils peuvent être exécutés dans des processus séparés, voire être déployés sur des machines distinctes (hétérogénéité). Chaque composant dispose d'un ou plusieurs services. Les services sont caractérisés par des ports d'entrée ou de sortie. Concrètement,

un composant SALOME représente un code de calcul. Il propose plusieurs services qui sont équivalents à des fonctions C, des méthodes C++ ou Python ou encore des sous-routines Fortran. Seuls les composants peuvent être couplés dans SALOME en connectant les points d'entrées et de sorties (ports) des services. Plus d'informations sont disponibles sur le site web de la plateforme : <http://www.salome-platform.org/>

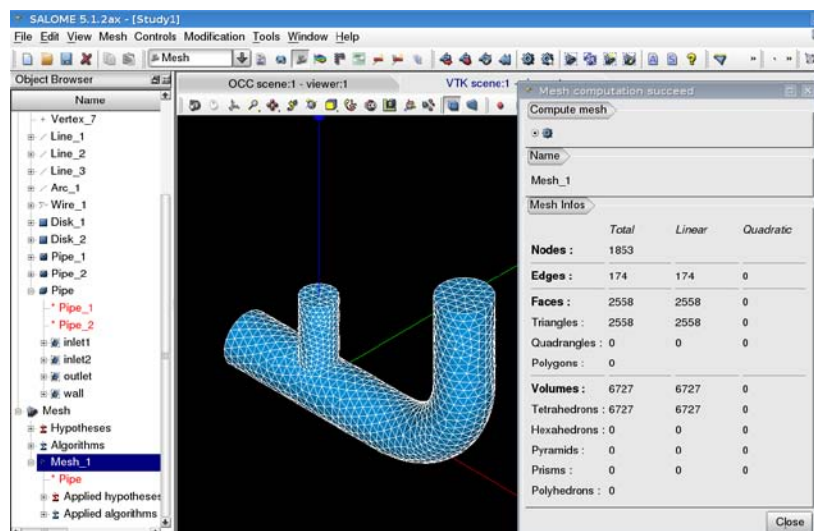
### III) SURVOL DE LA PLATEFORME VIA L'ETUDE D'UN CAS D'ECOLE

Le premier TD de cette session consiste à faire un rapide survol de l'utilisation des principaux modules de SALOME au travers d'un cas d'école : l'étude d'un mélange eau chaude – eau froide dans un tuyau :



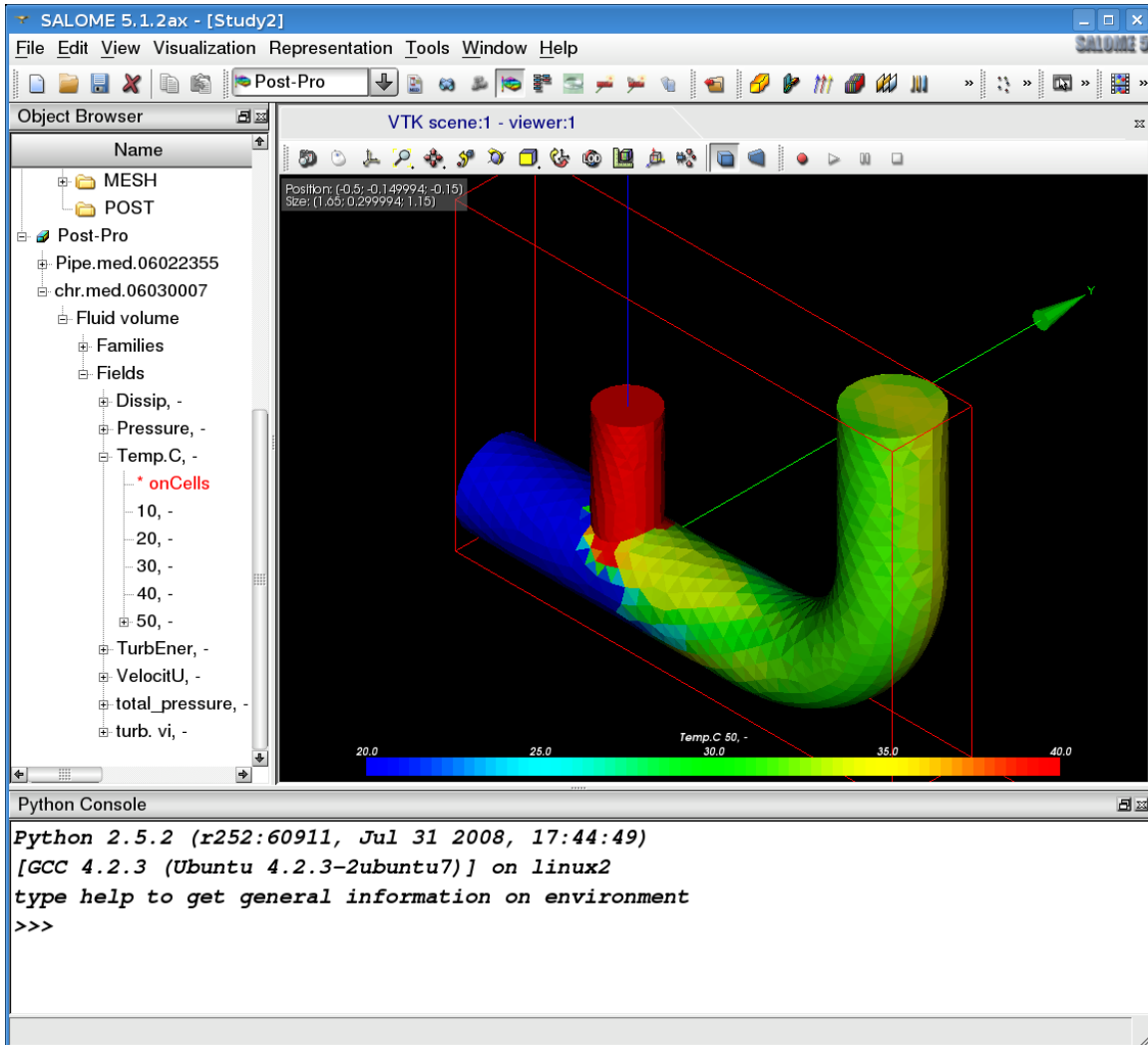
Dans un premier temps, nous allons utiliser le module GEOM afin de construire la géométrie de ce tuyau et d'en définir les frontières. Cette étape nous donne un rapide aperçu des principales fonctionnalités de ce module et une première prise en main de SALOME.

Ensuite, d'un simple clic, toujours dans l'interface graphique de SALOME, nous basculons sur un 2<sup>nd</sup> module : le module de maillage SMESH. Toujours en suivant pas à pas les instructions du tutorial, nous allons maintenant mettre en place un maillage tétraédrique sur notre tuyau :



Une fois le maillage réalisé et exporté vers un fichier MED, nous allons utiliser le code de CFD Code SATURNE (intégré dans SALOME) pour simuler l'écoulement. Un clic sur l'icône appropriée ouvre le module de Code SATURNE. Après l'importation du maillage, nous fournissons au code les paramètres physiques de notre étude.

Une fois le calcul effectué, nous exportons les résultats dans le dernier module que nous allons utiliser pour ce TD : le module de post processing et de visualisation VISU.



Ce TD de prise en main de la plateforme et d'utilisation des principaux modules se termine sur la découverte des options de visualisation. On notera la facilité avec laquelle il est possible d'enchaîner les modules afin de mener une étude complète. Il apparaît d'ors et déjà que les possibilités d'utilisation de chaque module sont nombreuses et que pour les utiliser efficacement une formation spécialisée est nécessaire.

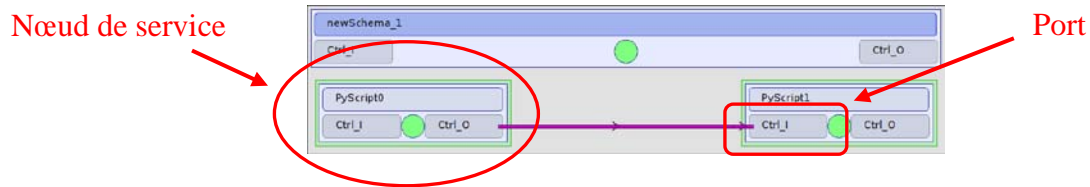


Notre première approche pratique sur YACS et la découverte de ses principales notions va se faire en programmant le calcul de Pi suivant l'algorithme de Gauss-Legendre. La première notion à connaître est celle de nœud de service. Un nœud de service représente une fonction de calcul particulière comme par exemple la simple addition de deux entiers ou bien un gros code de calcul. Il existe deux types de nœuds :

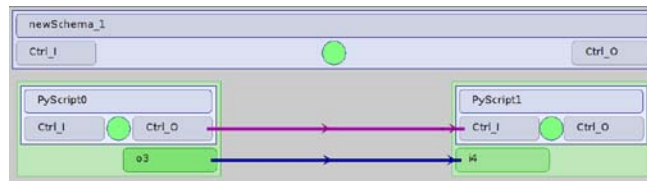
- ↳ des scripts python exécutés dans un processus SALOME
- ↳ des codes C++ ou pythons exécutés dans un container SALOME (un container est une entité SALOME permettant de regrouper plusieurs processus)

A la notion de nœud est associée celle de port. Un port représente une interface entre un nœud et l'extérieur. Il existe trois types de ports :

- ↳ les ports de contrôle (workflow) : utilisés pour mettre des contraintes sur l'enchainements de l'exécution de nœuds. Un nœud élémentaire dispose d'un port de contrôle entrant et d'un port sortant. Un nœud qui a son port de contrôle d'entrée connecté au port de sortie d'un autre nœud ne sera exécuté que lorsque ce deuxième nœud sera terminé.



- ↳ les ports de données (dataflow) : utilisés pour définir les données qui seront fournies au nœud au départ de son exécution et les données qui seront produites à la fin de son exécution



- ↳ les ports datastreams : utilisés pour l'échange de données pendant l'exécution



Pour faire l'analogie avec PrePALM, un nœud correspond à une unité, un container à un block, les ports d'entrées et de sorties correspondent aux plots d'entrées et de sorties des unités tandis que l'effet des ports datastream est comparable aux procédures PALM\_Put et PALM\_Get. L'établissement d'une communication entre les différents ports se fait comme avec PrePALM, en reliant les deux ports concernés. On peut avoir du 1->N, du N->1 ou du N->M.

Dans ce premier exercice, nous appréhendons le premier type de nœud (script python) en utilisant des services python déjà disponibles sous SALOME, (associé aux ports de contrôles)?. Cette première utilisation de YACS permet de nous familiariser avec le lancement des codes et la supervision du calcul (par exemple, la possibilité d'avancer pas à pas).

Un aspect important à aborder est celui de buffer (zone mémoire dans laquelle sont stockés les objets à échanger). Alors que dans PALM, il est possible de réaliser un contrôle fin du buffer (par exemple, il est possible d'effacer des objets en cours de simulation), aucune fonctionnalité de gestion de ce type n'est implémentée dans YACS. En effet, tous les objets échangés sont stockés en mémoire et sont donc accessibles tout au long de la simulation mais cet espace peut vite saturer dans le cas de nombreux échanges de gros objets.

Nous voyons de plus les limites de l'interface graphique puisque une fois que sont inclus quelques nœuds et quelques liens symbolisant les communications entre les ports, la vision du schéma d'ensemble est très surchargée.

## **V) UTILISATION DE YACSGEN**

Nous allons maintenant prendre en main l'outil YACSGEN et réaliser le même calcul de Pi que précédemment. YACSGEN est une interface python qui permet de fabriquer un composant SALOME automatiquement à partir d'un code extérieur à SALOME.

YACSGEN fabrique un fichier XML qui donne une description synthétique des composants (chaque composant pouvant inclure un ou plusieurs services). Il est possible de créer plusieurs types de composants (C/C++, Fortran 77, Python). Par exemple, si on veut interfacier un code écrit en Fortran 77, il faudra écrire un script Python qui contiendra des instructions spécifiques (écrites en Fortran 77). Ces instructions permettront de créer les services associés au code en question et notamment de mettre en place les différents ports (datastream ou autres).

Une fois ce programme Python compilé par YACSGEN, on pourra l'importer dans YACS et l'utiliser comme n'importe quel composant SALOME ; comme dans l'exemple précédent. Cet import se fait en chargeant le fichier XML précédemment généré. Avant de faire l'import, il est possible d'éditer ce fichier et d'y apporter des modifications. Ce qui suppose à ce point de maîtriser le Fortran 77, le python et le format XML ; en plus des outils SALOME et YACS.

Pour cet exercice, nous disposons d'un squelette écrit en Python dans lequel nous allons ajouter des instructions C++ afin de créer nos services et nos ports (dataflow uniquement pour le moment) qui seront utilisés dans YACS. Une fois ces services importés, on peut alors réaliser le chainage comme dans le premier cas et faire le calcul de Pi. Il est important de noter que pour faire du chainage de codes, on utilise les ports dataflow et

workflow et donc il n'y a aucune instrumentation de code à réaliser (contrairement à PALM où quelque soit le type de couplage, il y aura toujours une instrumentation à mettre en place).

Pour information, voici le script Python que nous devons compléter avec nos instructions C++ :

```
import os
from module_generator import Generator,Module,Service
from module_generator import CPPComponent

prerequis_file="/local/home/salome/SALOME5/V5_1_2_formation/prerequis-
V5_1_2_formation.sh"

kernel_root_dir=os.environ["KERNEL_ROOT_DIR"]
gui_root_dir=os.environ["GUI_ROOT_DIR"]
yacs_root_dir=os.environ["YACS_ROOT_DIR"]

context={'update':1,
         "makeflags":"-j2",
         "prerequisites":prerequis_file,
         "kernel":kernel_root_dir
         }

cwd=os.getcwd()

# PUT HERE DEFINITIONS OF THE COMPONENTS AND THE SERVICES

g=Generator(Module("ex2_cpp_module",components=[],prefix="/ex2_cpp_module_i
nstall"),context)
g.generate()
g.bootstrap()
g.configure()
g.make()
g.install()
g.make_appli("ex2_cpp_appli",
             restrict=["KERNEL","GUI","YACS"],
             altmodules={"GUI":gui_root_dir,
                       "YACS":yacs_root_dir})
```

Les instructions à ajouter pour créer notre composant, les services et les ports associés étaient :

```
c1=CPPComponent("Data",services=[
    Service("Input_Data",
        inport=[],
        outport=[("a0", "double"),
                 ("b0", "double"),
                 ("t0", "double"),
                 ("p0", "double")],
        body=body_input_data,
    ),
    Service("Save_Data",
        inport=[("an_1_in", "double"),
                ("bn_1_in", "double"),
                ("tn_1_in", "double"),
                ("pn_1_in", "double")],
        outport=[("an_1_out", "double"),
                 ("bn_1_out", "double"),
                 ("tn_1_out", "double"),
                 ("pn_1_out", "double")],
        body=body_save_data,
    ),
],
)
```

Un second exercice sur l'utilisation de YACSGEN et de YACS consistera cette fois à réaliser un couplage fort entre différents codes. Dans notre script Python (qui sera ensuite compilé par YACSGEN), nous allons inclure des ports datastream (seul moyen de faire du couplage fort) définis à partir de ports CALCIUM (ancien coupleur EDF ressemblant à PALM). En fait, pour réaliser du couplage fort dans YACS, les développeurs utilisent les fonctionnalités CALCIUM. L'avantage est que cette architecture est robuste et a fait ses preuves.

Cette fois, il est nécessaire de procéder à une instrumentation des codes. Cette phase aura été au préalable réalisée par les formateurs. Cette instrumentation correspond à des appels de fonctions CALCIUM dont on peut facilement faire l'analogie avec les PALM\_Put et PALM\_Get de PALM. L'instrumentation des codes se réduit aux appels d'envoi et de réception d'objets.

Le principal inconvénient lié à l'utilisation des ports CALCIUM est qu'il n'est pas possible de manipuler des objets de types dérivés (structures) mais uniquement les types de base ; ou encore d'envoyer des sous objets (envoyer à un code qu'une partie d'un gros tableau). Il n'est pas possible non plus de réaliser des interpolations sur les champs échangés en datastream car ces opérations ne s'appliquent que sur les objets de type MED.

Enfin, la manipulation des objets eux mêmes ne semble pas possible contrairement à ce que propose PALM au travers des diverses fonctions telles que des fonctions de conversion de dates ou encore des fonctions permettant de faire transiter des objets dynamiques (c'est-à-dire dont la taille peut varier au cours de la simulation).

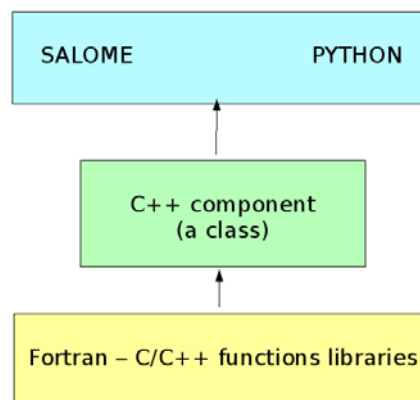
Concernant le couplage de codes en massivement parallèle, YACS accuse un sérieux retard sur PALM. Alors que pour PALM, les communications parallèles sont pleinement implémentées (redistribution automatique de données et fonctionnalité de localisation), ce n'est pas le cas pour YACS. A l'heure actuelle, le seul moyen de faire transiter un objet décomposé sur plusieurs processeurs et de regrouper toutes les parties de l'objet sur un unique processeur, de l'envoyer vers un processeur du code qui le réclame puis de redistribuer l'information entre les processeurs.

Nous devons ensuite faire un exercice qui nous aurait permis de manipuler une fonctionnalité très intéressante de SALOME, à savoir le lancement d'un couplage sur un réseau de machines hétérogènes. Faute de temps, nous n'avons pas pu aborder cette partie de la formation.

## **VI) UTILISATION DE HXX2SALOME**

Tout comme YACSGEN, HXX2SALOME est un outil (développé par le CEA) qui permet de générer automatiquement des composants SALOME. Au cours des exercices suivants nous allons apprendre à créer des composants C++ SALOME et utiliser le format MED (standard EDF/CEA, qui permet d'envoyer des objets d'un code à un autre).

Nous l'avons vu précédemment, un composant SALOME représente un code de calcul. Il propose plusieurs services qui sont équivalents à des fonctions C, des méthodes Python ou C++ ou encore des sous-routines Fortran. Seuls les composants peuvent être couplés dans SALOME en connectant les ports d'entrés et les ports de sortie des services. Avant de pouvoir générer un composant SALOME avec HXX2SALOME, il faut d'abord passer par la création d'un composant C++ :



Ainsi, pour transformer un code source écrit en Fortran, C ou C++ en composant C++, il faut :

- ↪ utiliser un outil nommé "SA\_new\_cpp\_component" qui va créer une arborescence comprenant des Makefiles,
- ↪ utiliser une interface Python afin de créer le composant C++ en question,
- ↪ utiliser ghxx2salome (interface graphique de HXX2SALOME) afin de construire le composant SALOME correspondant.

Dans cet exercice, le couplage se fait donc en dataflow (aucune intrusion dans les codes à coupler) et les objets échangés sont de type MED (standard EDF ; cf § suivant). Comme nous utilisons ce type bien précis, il est alors possible de faire des interpolations d'un maillage à un autre sur les champs MED.

Le module SMESH de SALOME permet de visualiser un maillage qui aura été au préalable sauvegardé au format MED. En plus de faire de la visualisation de maillage, ce module permet de vérifier ses propriétés comme par exemple le nombre d'éléments total. La création d'un maillage peut se faire de différentes façon. Ici, nous plaçons nos instructions dans un script Python.

La principale restriction quand à l'utilisation de HXX2SALOME est qu'il n'est pas possible de mettre en place des ports datastream dans les services. Il n'est donc pas possible ici de faire du couplage fort.

## ***VI ) INTRODUCTION AU FORMAT MED***

La création du format MED résulte de deux besoins :

- ↪ pouvoir importer et exporter des données d'un code à un autre dans un format commun
- ↪ pouvoir appliquer des fonctionnalités de maillage, interpolation spatiale et de visualisation.

La première version de ce modèle d'échange fonctionnait uniquement avec des données sous forme de fichiers. Ensuite est venue la version mémoire MEDMEM puis la version parallèle PARA-MEDMEM. Ce système repose sur le standard de fichiers HDF5 auquel ont été ajoutées des fonctionnalités :

- ↪ deux types de représentations de maillages : connectivité descendante et nodale,
- ↪ différents types de maillages : non structurés, cartésiens, polyèdres, ...

Grâce à ce standard, il est possible de faire des opérations d'interpolation spatiale sur les champs échangés entre des codes qui suivent le format de données MED sous SALOME.