Coupling of CTF with ATHLET and DYN3D on the SALOME platform

Yann Périn (GRS), Javier Jimenez (KIT), Oleg Murarius (INRNE) , Kiril Velkov (GRS)

CTF User Group Meeting

PSI, 2-3 June 2016

Content

- Description of the NURESAFE Project
- Description of the Salomé platform
- Code integration and coupling on the platform
- CTF integration and coupling on the platform
- Conclusions

Description of the NURESAFE Project

- NURESAFE = NUCLEAR REACTOR SAFETY SIMULATION PLATFORM
- The NURESAFE project addresses engineering aspects of nuclear safety, especially those relative to design basis accidents (DBA). In this respect, the best simulation software are needed to justify the design of reactor protection systems and measures taken to prevent and control accidents.
- The NURESAFE project addresses safety of light water reactors which will represent the major part of fleets in the world along the whole 21st century.
- The first objective of NURESAFE is to deliver to European stakeholders a reliable software capacity usable for safety analysis needs and to develop a high level of expertise in the proper use of the most recent simulation tools.
- This software capacity will be based on the NURESIM simulation platform created during FP6 NURESIM project and developed during FP7 NURISP project which achieved its goal by making available an integrated set of software at the state of the art.
- The objectives under the work-program are to develop practical applications usable for safety analysis or operation and design and to expand the use of the NURESIM platform.
- The main outcome of NURESAFE will be the delivery of multiphysics and fully integrated applications.

Description of the NURESAFE Project

- Organized within EU‘s 7th Framework program
- 50% founded from the European Commission and 50% self-financing from partners
- 23 institutions from 14 countries involved
- Project start: 01/01/2013
- Project duration: 36 months
- Project end: 31/12/2015

Description of the NURESAFE Project

- SP1: Multiphysics applications involving core physics

➔ CTF applied to PWR, BWR and VVER

- SP2: Multiscale analysis of core thermal-hydraulics from DNS to subchannel modeling

- SP3: Multiscale and multiphysics applications of thermalhydraulics

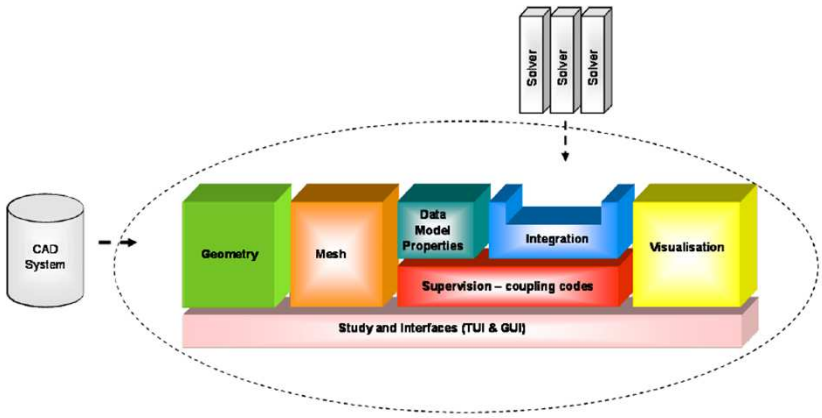
➔ CTF applied to BWR (ATHLET/CTF coupling without NK)

- SP4: Platform

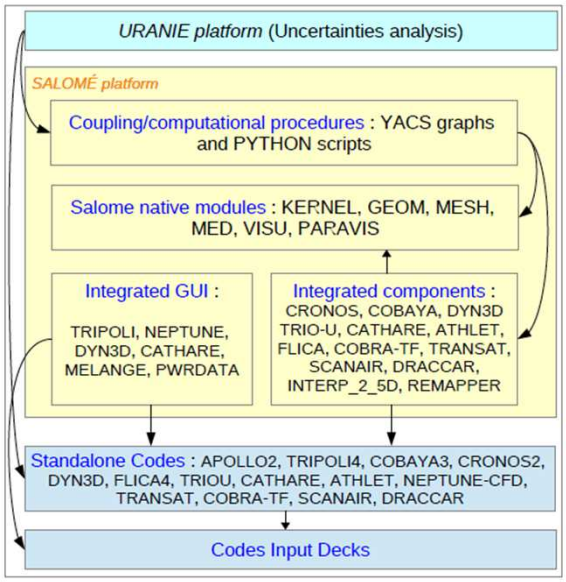
Description of the Salomé (=NURESIM) Platform

- Pre-processing, post-processing, code integration and code coupling

- Open-source project : www.salome-platform.org



Description of the Salomé (=NURESIM) Platform



Integration of codes on the platform

- C++ interface (API)
- Code as static libraries
- SalomeTool (SAT) is a suite of commands in Python which mutualizes the maintenance tasks of the Salomé platform and its applications.
  - Management of multiple source repositories (CVS, SVN, GIT, . . .)
  - **Compilation of modules** and prerequisites
  - **Module generation** (wrapping of YACSGEN module)
  - Standardised and automatic testing procedures
  - Creation and installation of packages for various OS

Coupling of codes on the platform

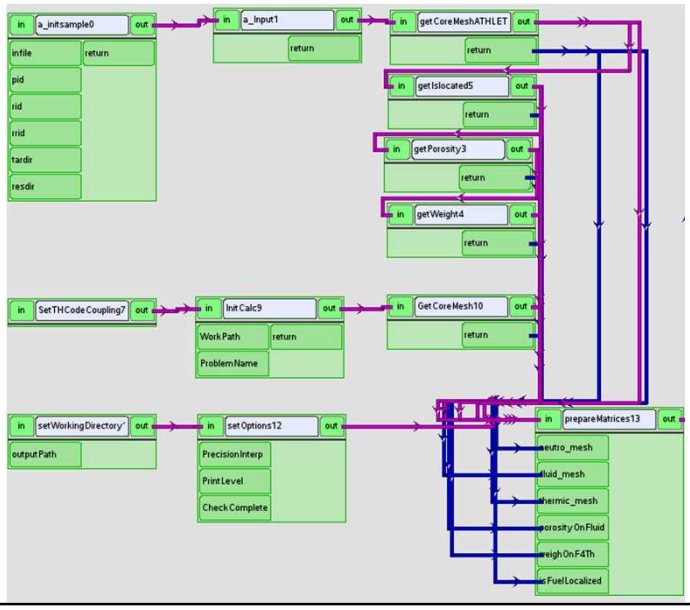
- Use of the so-called MEDCoupling format for data visualization and exchange
  - Mesh (geometry)
  - Fields (data)

- Interpolation tools are included on the platform
  - INTERP_2_5D component (e.g. DYN3D coupling):
    - Developed for FLICA-CRONOS coupling
    - Can be used for any TH/NK applications
  - REMAPPER component (e.g. ATHLET Coupling):
    - Less specific
    - More versatile

Building a computation scheme with YACS

Initialization graph

- ATHLET
- DYN3D
- INTERP_2_5D



Building a computation scheme with Python

- Example of a Steady-state case

DYN3D steady-state calculation

Data interpolation and transfer D3D -> ATHLET

ATHLET steady-state calculation

Data interpolation and transfer ATHLET -> D3D

```
i=1
status=0
while(status==0):
  # DYN3D iteration
  status=d3d.ComputeSteadyStateStep()
  if status>0:
      #
      print "Error in DYN3D steady-state"
      break
  # Get power distribution from DYN3D
  power=d3d.GetCorePowerField(0)
  ptot=d3d.GetTotalPower()
  print "Power:",ptot
  print "Keff:",str(d3d.GetKeff())
  # Interpolate field
  (ffluid,ffuel)=i25d.projectPower(power)
  a30.setInputMEDField("pow",ffuel)
  # ATHLET iteration
  aret=a30.a_Steady()
  #
  # Transfer the feedback data
  fteco=a30.getOutputMEDField("fuel_temperature")
  teco=i25d.projectThField(fteco)
  d3d.SetFuelTemperatureField(teco)
  #
  fpbor=a30.getOutputMEDField("boron_concentration")
  pbor=i25d.projectFluidField(fpbor)
  d3d.SetBoronConcentrationField(pbor)
  #
  fdmod=a30.getOutputMEDField("moderator_density")
  dmod=i25d.projectFluidField(fdmod)
  d3d.SetModeratorDensityField(dmod)
  #
  i+=1
  #
  pass
```

Description of the CTF API – General control

Services	Description
bool initialize()	Initializes the code variables Reads input Returns true
void setInput(const char* name);	Set input path name: path to input The path cannot be longer than 100 characters If not set, the code assumes a deck.inp input in the working directory
void transientMode()	Initializes transient by changing RTWP parameter to 1.0 and resetting forcing tables
void terminate()	Finalizes code run Closes output files

Description of the CTF API – Simulation control

Services	Description
double presentTime() const	Returns present time in CTF simulation
double computeTimeStep() const	Gives the next proposed time step size in CTF Returns time step size
void validateTimeStep()	Finalizes time step calculation in CTF
bool isStationary() const	Checks if steady-state is reached in CTF Returns true if stationary
void abortTimeStep()	Resets variables to previous time step value

Description of the CTF API – Mesh control

Services	Description
void SetCoreMeshRotation (double rot)	Set rotation of 3D core mesh rot: rotation in degree Must be called before genMeshCTF
void SetCoreMeshTranslation (double xtrans, double ytrans, double ztrans);	Set translation of 3D core mesh in Cartesian geometry xtrans: translation in x direction ytrans: translation in y direction ztrans: translation in z direction Must be called before genMeshCTF
void genMeshCTF(int geom, int count_of_assemblies)	Generates 3D core mesh for different types of modelling geom: 0 FUEL CENTERED RECTANGULAR GEOMETRY 1 FUEL CENTERED HEXAGONAL_GEOMETRY 2 HEX_COOLCENT_FINEMESH_GEOMETRY 3 HEX_RODCENT_FINEMESH_GEOMETRY count_of_assemblies: number of modelled assemblies (dummy if geom!=3)

CTF Meshing

- Two different meshings (necessary for interpolation component)
  - Fluid Meshing
  - Fuel Rod Meshing
- Fluid meshing depends on the geometry
  - Quadratic (/Rectangular) geometry
  - Hexagonal geometry
  - Hexagonal Pin-by-pin geometry
    - Fuel centered
    - Coolant centered

CTF Meshing

- Quadratic meshing is fully automated
  - If rod multiplication factor = 1.0 → Pin by pin (rod mesh size != channel mesh size)
  - Else → Assembly wise mesh (rod mesh size == channel mesh size)
  - If channel map == rod map → rod centered mesh
  - Else → channel centered
- Possibility to use refined meshing
  - „Trick“ in rod/channel maps
  - Only possible for rod centered mesh

CTF Meshing

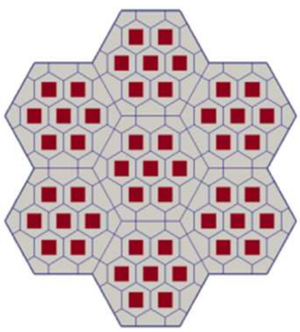
- Refined meshing
  - Channel map

```
*Card 17.4 - Channel Map
*  R  R  R  R  R  *
   1  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
*  R  F  F  F  R  *
   1  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
*  R  F  P  P  P  P  P  P  P  P  P  P  P  P  P  P  P  P  P  F  R
   1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  0  0
   0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  0  0
   0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  0  0
   0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  0  0
   0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  0  0
   0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  0  0
   0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  0  0
   0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  0  0
   0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  0  0
   0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  0  0
   0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  0  0
   0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  0  0
   0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  0  0
   0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  0  0
   0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  0  0
   0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  0  0
   0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
*  R  F  F  F  R  *
   1  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
*  R  R  R  R  R  *
   1  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

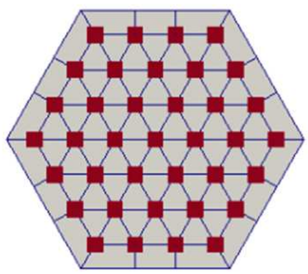
CTF Meshing

- Hexagonal Pin-by-pin geometry
  - Fuel centered
  - Coolant centered
- Generation tool is separated from the cartesian geometry one

Developed with INRNE



Developed with KIT



Description of the CTF API – Mesh control

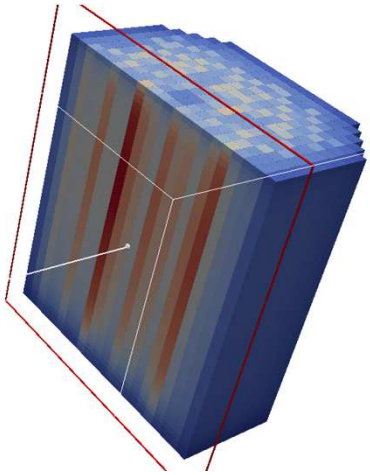
Services	Description
MEDCouplingUMesh* getFluidMeshCTF() const	Provides the 3D mesh of the core fluid structure Returns a MEDCouplingUMesh object
MEDCouplingUMesh* getRodMeshCTF() const	Provides the 3D mesh of the core rods structure Returns a MEDCouplingUMesh object
MEDCouplingUMesh* getInletMeshCTF() const	Provides the 2D mesh of the core inlet Returns a MEDCouplingUMesh object
MEDCouplingUMesh* getOutletMeshCTF() const	Provides the 2D mesh of the core outlet Returns a MEDCouplingUMesh object

Description of the CTF API – Fields control

Services	Description
MEDCouplingFieldDouble* getOutputMEDField(const std::string& name) const	Provides the MEDCoupling field for a given name name: name of field Accepted names: “moderator_density” “moderator_temperature” “fuel_temperature” “porosity” “weight” “islocated” Returns a MEDCouplingFieldDouble object
void setInputMEDField(const std::string& name, const MEDCouplingFieldDouble* afield)	Sets the value of a given MEDCoupling field in CTF name: name of the field Accepted names: “power” “inlet_massflow” “inlet_enthalpy” “outlet_pressure” afield: MEDCoupling field object

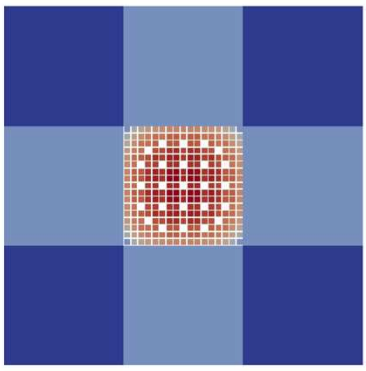
CTF Fields

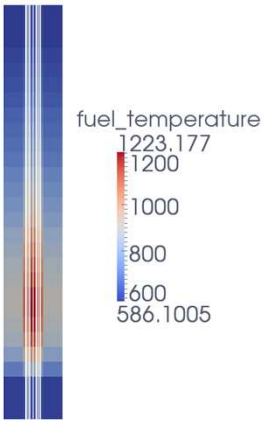
- The function getOutputMEDField delivers the TH feedbacks in the core
- TH feedbacks fields using the fluid meshing
  - moderator_density
  - moderator_temperature
  - boron_concentration
- TH feedbacks fields using the rod meshing
  - fuel_temperature
  - power
- Fields extraction uses the ctf_coupling_interface module



CTF Fields

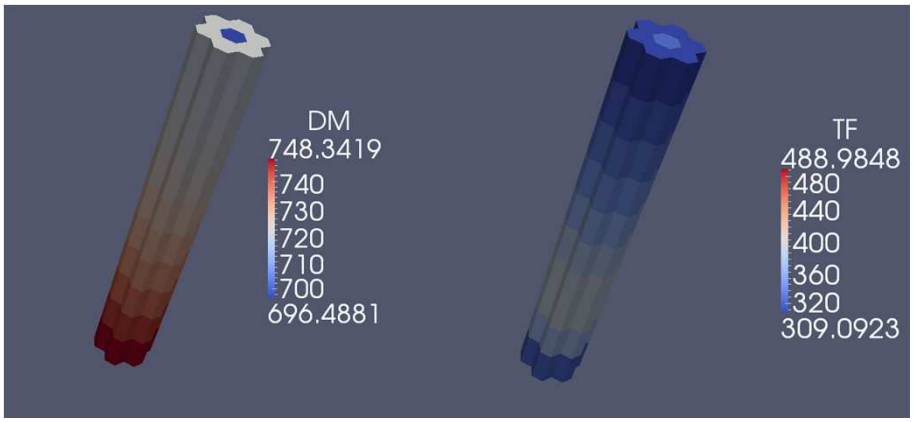
- Example of an hybrid meshing
  - 3y3 minicore with refined mesh in the center fuel assembly
  - Coupled with DYN3D (with pin reconstruction)





CTF Fields

- CTF / DYN3D Coupling for Hexagonal geometry
  - 7 Assemblies mini-core steady-state results



Description of the CTF API – Fields control

Services	Description
void writeFieldinVTK(const char* name, const MEDCouplingFieldDouble* afield)	Write a given MEDCoupling field in a VTK file (.vtu) name: name of field afield: MEDCoupling field object
double AverageValue(const MEDCouplingFieldDouble* afield) const	Gives the average value of a field object afield: MEDCoupling field object Returns the average value

TH/TH coupling

- A one-way coupling with domain overlaping with ATHLET was developed
- For this coupling at core inlet/outlet 2D Inlet/outlet meshes are created
  - getInletMeshCTF
  - getOutletMeshCTF
- The following field fields are accepted
  - „inlet_temperature“, 2D field from ATHLET
  - „inlet_massflow“, 2D field from ATHLET
  - „outlet_pressure“, 2D field from ATHLET

  } Interpolation is done here using the REMAPPER library
- Explicit time coupling

Conclusions

- Coupled applications of CTF in the NURESAFE project
  - Full core assembly-wise simulation with the ATHLET-DYN3D-CTF coupled system
    - PWR Main Steam Line Break (Zion reactor) → See next presentation
    - BWR Turbine Trip (Peach Bottom) → See next presentation
  - Pin-by-pin coupled simulation with CTF-DYN3D (PWR assembly)
  - Pin-by-pin coupled simulation with CTF-COBAYA (hexagonal assembly)
  - ATHLET/CTF multi-scale TH simulations of the Oskarshamn-2 core