NURESAFE WP3.3
Multiscale BWR Thermal-Hydraulics

Status of KIT Contributions to WP3.3

J. Jimenez, V. Sanchez

Presented by J. Jimenez

javier.jimenez@kit.edu
or
victor.sanchez@kit.edu

Outline

- Short review of work done within WP3.3
- Description of ATHLET SALOME component
- Description of COBRA-TF SALOME component
- Description of the multi-scale coupling ATHLET/COBRA-TF
- Application to BWR-ATWS
- Conclusion & Outlook

Outline

- Short review of work done within WP3.3
- Description of ATHLET SALOME component
- Description of COBRA-TF SALOME component
- Description of the multi-scale coupling ATHLET/COBRA-TF
- Application to BWR-ATWS
- Conclusion & Outlook

Validation of ATHLET using the BWR NUPEC BFBT

- **This task is completed.**
- **Paper published in Nuclear Engineering and Design**

Nuclear Engineering and Design 288 (2015) 183–194

Contents lists available at ScienceDirect

Nuclear Engineering and Design

journal homepage: www.elsevier.com/locate/nucengdes

ELSEVIER

Validation of the thermal-hydraulic system code ATHLET based on selected pressure drop and void fraction BFBT tests

Valentino Di Marcello*, Javier Jimenez Escalante, Victor Sanchez Espinoza

Karlsruhe Institute of Technology, Institute for Neutron Physics and Reactor Technology, Hermann-von-Helmholtz-Platz 1, D-76344 Eggenstein-Leopoldshafen, Germany

HIGHLIGHTS

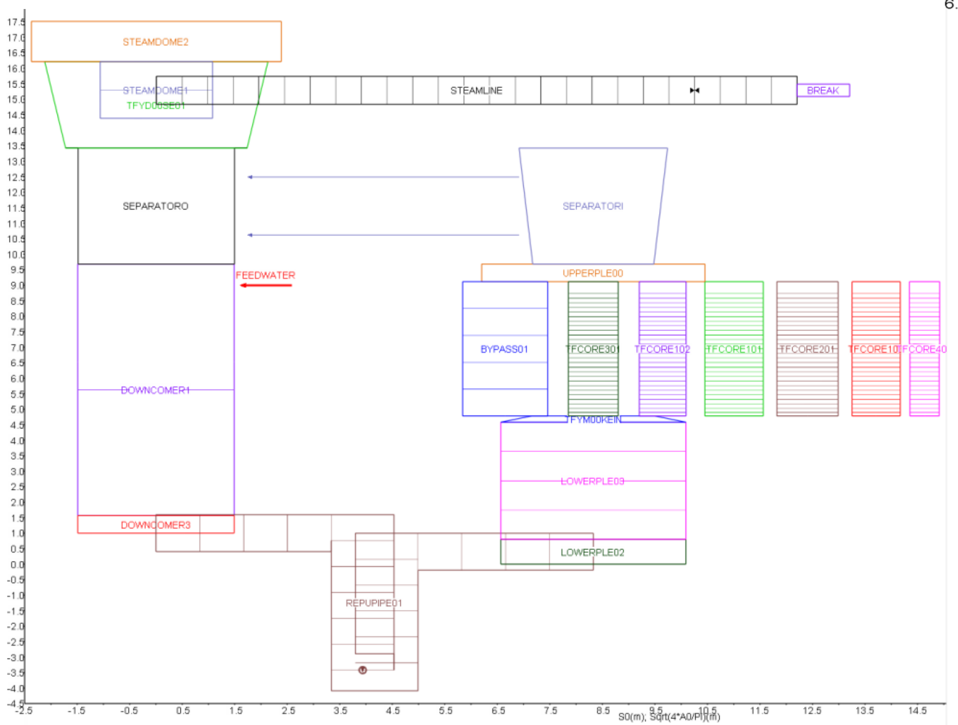
- Simulation of BFBT-BWR steady-state and transient tests with ATHLET.
- Validation of thermal-hydraulic models based on pressure drops and void fraction measurements.
- TRACE system code is used for the comparative study.
- Predictions result in a good agreement with the experiments.
- Discrepancies are smaller or comparable with respect to the measurements uncertainty.

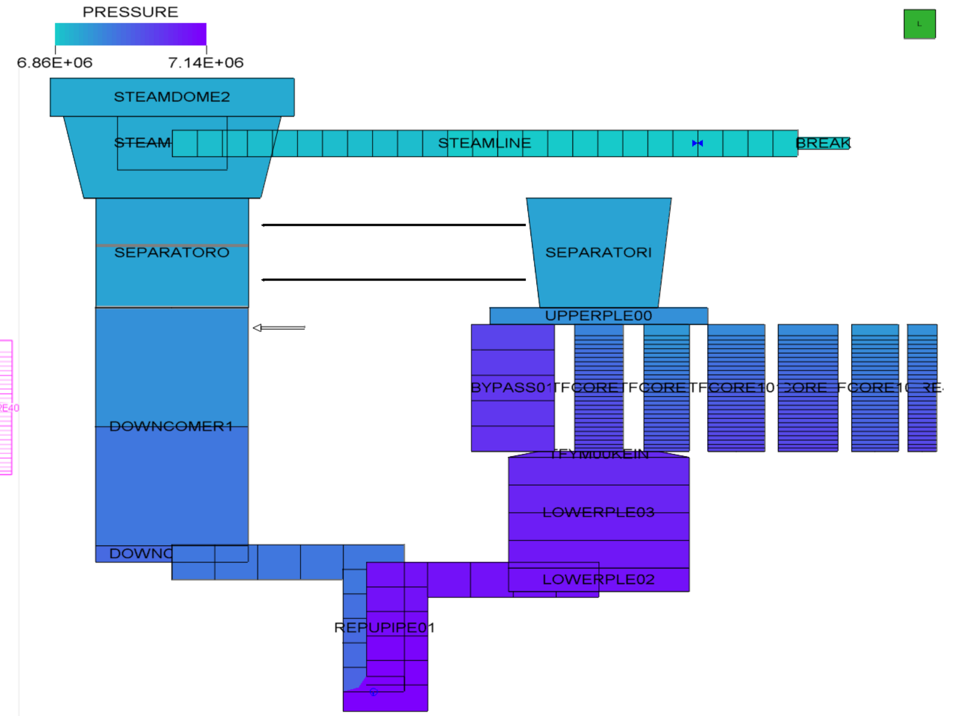
http://dx.doi.org/10.1016/j.nucengdes.2015.04.003

ATHLET model for O-2 using 6 sectors

- **This task is completed (collaboration with GRS).**
- **Using ATHLET Mod3.0 Cycle A for the WP1.3 and WP3.3 task**

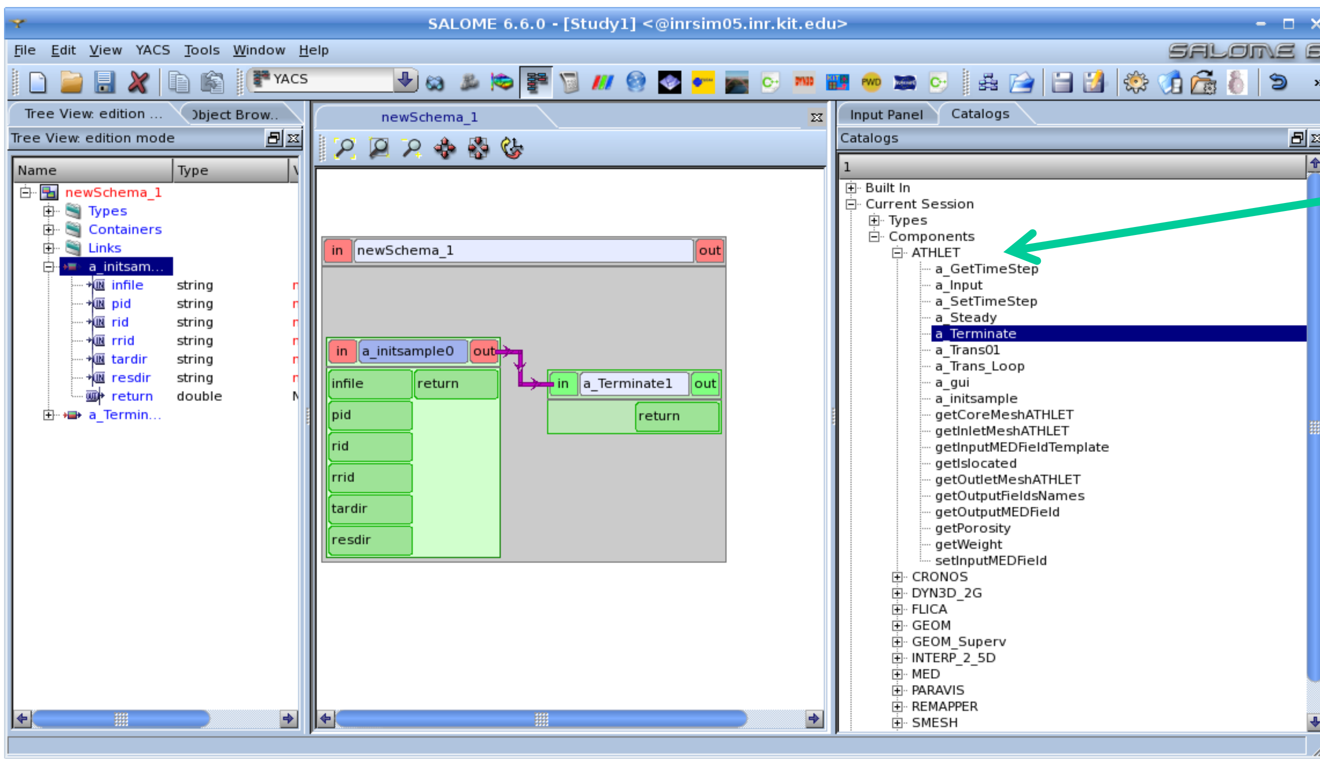
Model contains: Downcomer, Recirculation Loop and Pump, Lower Plenum, Steam Separator, Steam Dome, Core Model, Core Bypass, Steam Line.





Preliminary ATHLET results at HFP

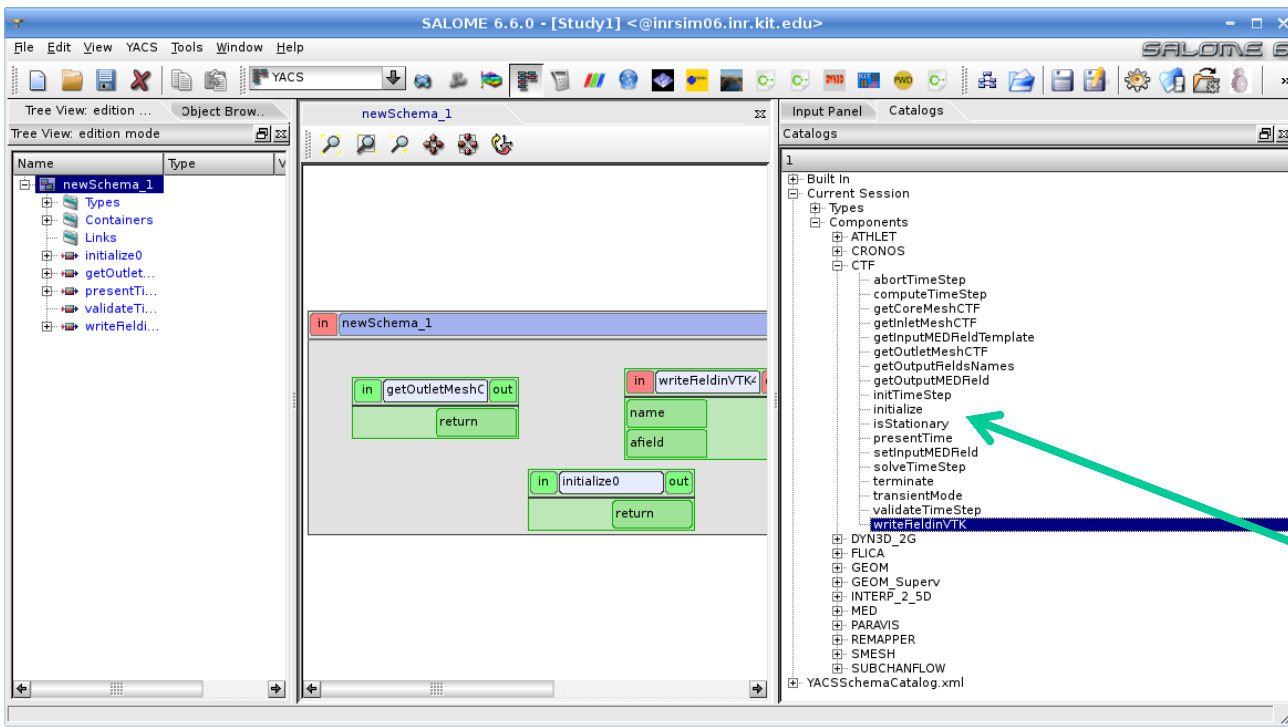
- **Comparison of values against measured data.**
  - Good agreement between the ATHLET model and the measured data.
- **Results were reported in the 4th SP3 meeting and in the General Seminar which took place in Budapest.**

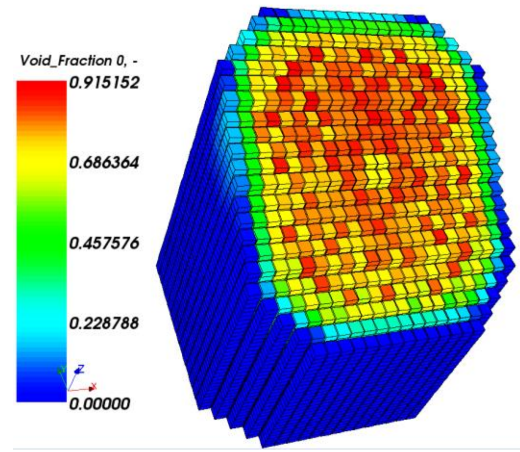


1st Preliminary version of ATHLET under SALOME available 30.10.2014

O2 Modeling with subchannel codes

- This task is completed.
- Code versus measured data comparison.
- Results were reported in the 4th SP3 meeting and in the General Seminar which took place in Budapest.





1st Preliminary version of CTF under SALOME available 30.10.2014

Summary of recent updates since 5th SP3

- **COBRA-TF and ATHLET API code and documentation were delivered and uploaded in the svn NURESAFE repository (deliverables by GRS).**
- **Some sample python coupling scripts are available.**
- **Still to define the type of ATWS to be run within WP1.3**
  - Big delay in the XS libraries delivery by KTH. No nuclear data still usable for the O2 core (DYN3D).
  - Option to change from O2 to PBTT was discussed during the last SP1 meeting.
  - A decision is to be taken by the WP1.3 leader (GRS) ASAP.
  - The delays within SP1 will not affect the SP3 work as the ATHLET/COBRA-TF coupling is already working and the input decks have been tested already.

Outline

- Short review of work done within WP3.3
- Description of ATHLET SALOME component
- Description of COBRA-TF SALOME component
- Description of the multi-scale coupling ATHLET/COBRA-TF
- Application to BWR-ATWS
- Conclusion & Outlook

Description of ATHLET SALOME component

- **The ATHLET version implemented on the Salomé platform is the current ATHLET release Version 3.0 Cycle a (See D11.15).**

- **The ATHLET API is made out of two C++ files: ATHLET30.hxx and ATHLET30.cxx. The public functions are divided into two groups: the ATHLET control functions and the coupling functions.**

- **Changes from standalone ATHLET input deck in order to run a coupled calculation:**
  - The NEUKIN3D part of the ATHLET input need to be present in order to get a 2D Meshing of the inlet/outlet.

ATHLET API control functions

Function	Description
double a_gui()	Launches ATHLET graphical interface which allows choosing the paths/input. Returns the CPU time taken by the function.
double a_initsample(const char* infile, const char* pid, const char* rid, const char* rrid, const char* tardir, const char* resdir)	Same goal as a_gui but through function parameters Returns the CPU time taken by the function.
double a_Input()	Reads input and initializes ATHLET. Returns the CPU time taken by the function.
double a_Steady()	Performs ATHLET steady-state iterations. Returns the CPU time taken by the function.
double a_Trans01()	Initializes ATHLET transient calculation. Returns the CPU time taken by the function.
double a_Trans_Loop(int MIZS, double TE)	Performs either MIZS time steps or TE seconds of transient. Returns the CPU time taken by the function.
double a_Terminate()	Finalizes ATHLET run. To be used at the end of the simulation. Returns the CPU time taken by the function.

ATHLET API coupling functions

Function	Description
double a_GetTimeStep()	Return the preferred size for the next time step.
ParaMEDMEM::MEDCouplingFieldDouble* getInputMEDFieldTemplate(const std::string& name) const	Returns a template of MEDCouplingFieldDouble of name “name”.
ParaMEDMEM::MEDCouplingFieldDouble* getOutputMEDField(const std::string& name) const	Returns a MEDCouplingFieldDouble of an ATHLET thermal-hydraulics feedback of name “name”. Only the names returned by getOutputFieldsNames are valid.
std::vector<std::string> getOutputFieldsNames() const	Returns the list of ATHLET thermal-hydraulics feedbacks that are available.
void setInputMEDField(const std::string& name, const ParaMEDMEM::MEDCouplingFieldDouble* afield)	Set the power distribution in ATHLET. Take a MEDCouplingFieldDouble as input. This MEDCouplingFieldDouble must follow the template returned by getInputMEDFieldTemplate.
ParaMEDMEM::MEDCouplingUMesh* getMeshATHLET() const	Returns the ATHLET 3D core meshing as a MEDCouplingUMesh object.
ParaMEDMEM::MEDCouplingFieldDouble* getPorosity() const	Returns the porosity MEDCouplingFieldDouble in the core for interpolation purposes. (Needed by INTERP_2_5D)
ParaMEDMEM::MEDCouplingFieldDouble* getWeight() const	Returns the weight MEDCouplingFieldDouble in the core for interpolation purposes. (Needed by INTERP_2_5D)
ParaMEDMEM::MEDCouplingFieldDouble* getIslocated() const	Returns the islocated MEDCouplingFieldDouble in the core for interpolation purposes. (Needed by INTERP_2_5D)

Outline

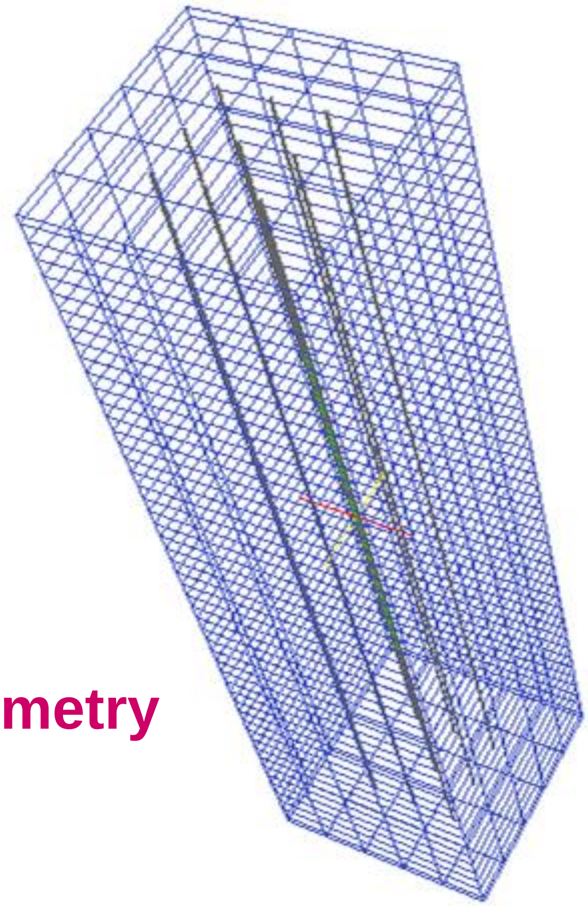
- Short review of work done within WP3.3
- Description of ATHLET SALOME component
- **Description of COBRA-TF SALOME component**
- Description of the multi-scale coupling ATHLET/COBRA-TF
- Application to BWR-ATWS
- Conclusion & Outlook

Description of the CTF API

- **The following functions are implemented for simulation control**
  - Initialize (calls init_tf, ctf_init, input…)
  - Terminate (calls post_trans)
  - presentTime (returns current simulation time)
  - computeTimeStep (calls timstp)
  - initTimeStep (sets delt)
  - solveTimeStep (calls prep3d, heat, outer_iteration, post3d)
  - validateTimeStep (calls post_step)
  - abortTimeStep (calls reset_trod and reset_flow)
  - isStationary (calls chk_converge)
  - transientMode (sets heat transfer time step ratio to 1)

CTF Meshing

- **Two different meshings (necessary for interpolation component)**
  - Fluid Meshing
  - Fuel Rod Meshing

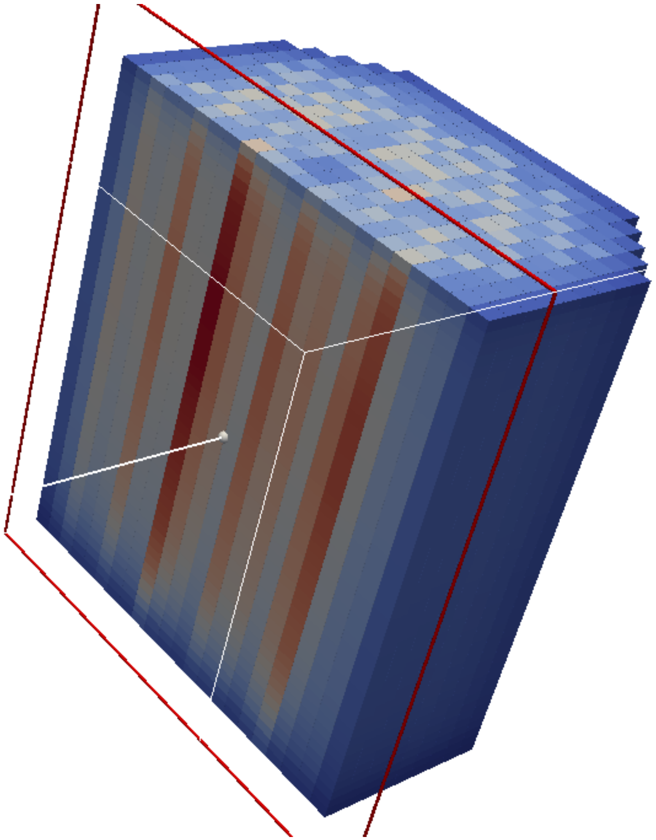


- **Fluid meshing depends on the geometry**
  - Quadratic (/Rectangular) geometry
  - Hexagonal geometry
  - Triangular geometry (in progress, needed in WP1.4)

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**

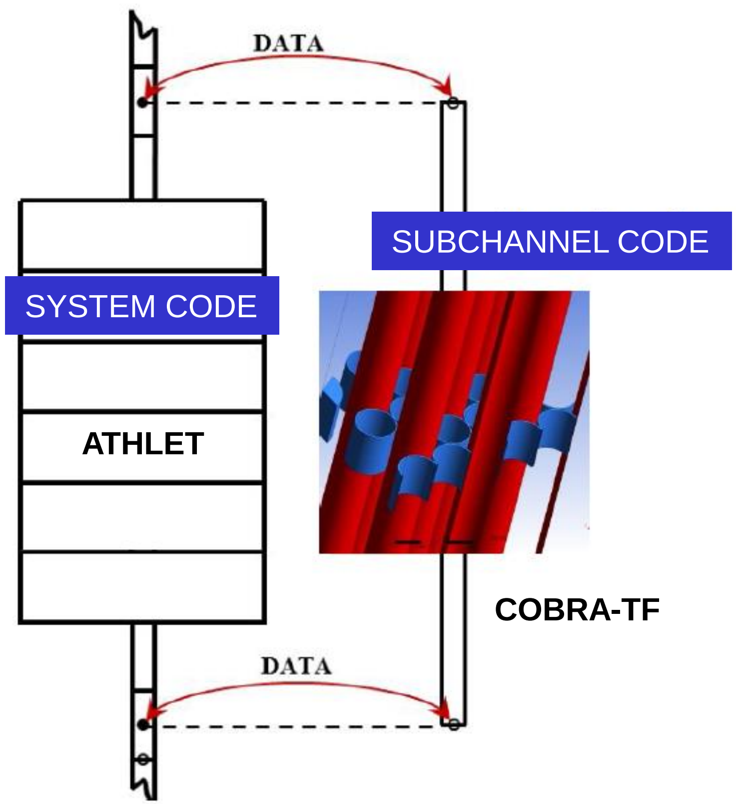
Outline

- Short review of work done within WP3.3
- Description of ATHLET SALOME component
- Description of COBRA-TF SALOME component
- **Description of the multi-scale coupling ATHLET/COBRA-TF**
- Application to BWR-ATWS
- Conclusion & Outlook

Review on coupling classification

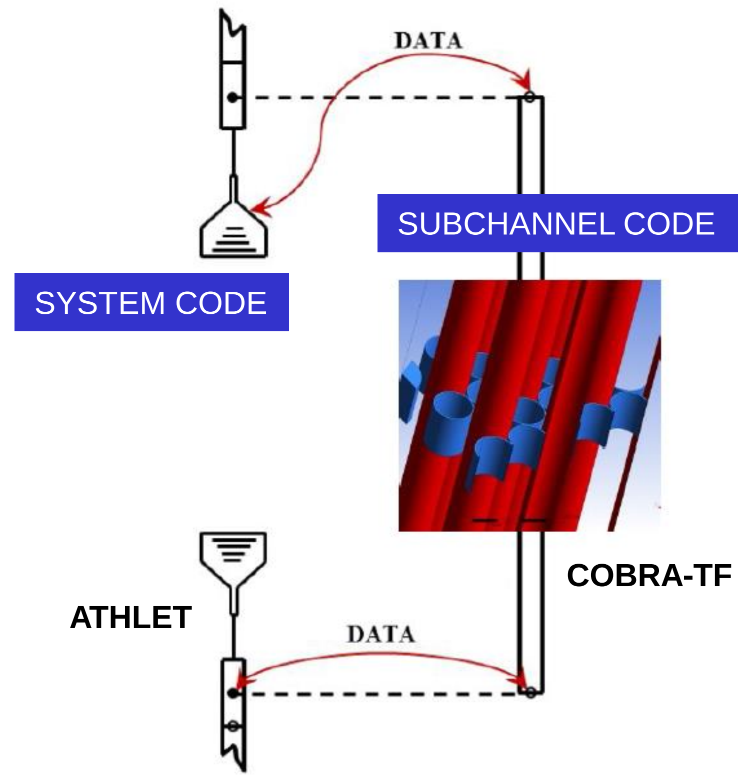
- Coupling of spatial domains

Overlapping domains



Domains are spatially superimposed to some extent

Non-overlapping domains



Domain is split into separate regions with well defined interfaces

Review on coupling classification

- **Synchronization**

Off-line Coupling

Codes run separately and sequentially. Results from one code are used as boundary or initial conditions for the other.

- **Simple** to implement and **no modifications** of the codes is requested;
- The information transfer is only "**one-way coupling**", **no feedback** is possible.

Case of ATHLET/COBRA-TF using MEDCoupling

In-line coupling

Codes run **concurrently** with a **continuous exchange** of information in both ways ("**two-way coupling"**)

Case of CATHARE/TRIO_U using ICOCO

- **Code Integration**

Internal coupling: Ad hoc solver to simultaneously solve the coupled system; Transfer internal memory

External coupling: Independent solvers are employed (coupling interface is needed).

Review on coupling classification

- **Numerical schemes (in-line coupling)**

Explicit coupling scheme

Time iteration is pure explicit, the codes take the minimum allowed time step for numerical stability and courant limit.

Implicit Coupling scheme

On each time step there is a inner iteration loop, convergence is achieved, allows for much bigger time step sizes.

TH/TH coupling

- **A one-way coupling with domain overlaping between ATHLET/COBRA-TF 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**

Coupling script (Part 1)

It can be found in:

https://www-svn-corpus.cea.fr/nuresafe/NURESIM/COUPLING_SCRIPTS

```
# Definition of the environment and libraries to be used

# Ressources of test base
ressourcedir=getenv("NURESAFE_TEST_DATA")

## COBRATF PARAMETERS
CTF_in    = ressourcedir + "/data/cobratf/" + casename + "/" + typecase
CTF_out   = getenv("PWD")
CTF_mesh  = getenv("PWD") + "/COBRATFMESH.med"
CTF_mesh2= getenv("PWD") + "/COBRATFSTRUCTURE.med"
system("ln -sf " + CTF_in + "/" + CTF_file + " deck.inp")
if path.exists(CTF_mesh): remove(CTF_mesh)
if path.exists(CTF_mesh2): remove(CTF_mesh2)

## ATHLET PARAMETERS
ATHLET_in    = ressourcedir + "/data/athlet/" + casename + "/" + typecase
ATHLET_out   = ATHLET_in + "/results"
ATHLET_mesh = getenv("PWD") + "/ATHLETMESH.med"
ATHLET_mesh2= getenv("PWD") + "/ATHLETSTRUCTURE.med"
# Create the output folder if not existing
shutil.rmtree(ATHLET_out,True)
makedirs(ATHLET_out)
if path.exists(ATHLET_mesh): remove(ATHLET_mesh)
if path.exists(ATHLET_mesh2): remove(ATHLET_mesh2)

t_start=time()
```

Coupling script (Part 2)

```
##################
# Init ctf
s1=ctf.initialize()
if hexagonal: ctf.genHexMeshCTF()
print "Initializing CTF" ,s1
print "-----------------------------------"
print "Initialization of COBRATF DONE"
print "-----------------------------------"

##################
# Init ATHLET
print ATHLET_in+"/"+ATHLET_file
s1=a30.a_initsample(ATHLET_in+"/"+ATHLET_file,casename,"run","",ATHLET_out,"")
s1=a30.a_Input()
print "Initializing ATHLET" ,s1
aret=a30.a_Steady()
a30.a_Trans01()
print "-----------------------------------"
print "Initialization of ATHLET DONE"
print "-----------------------------------"

##################
# Calculation
status=0
ttime=0.0 # If steady state calculation, ttime is always 0.0
Tend=10.0
stepsize=0.0
it=1
told=0.0
delta=1.0
```

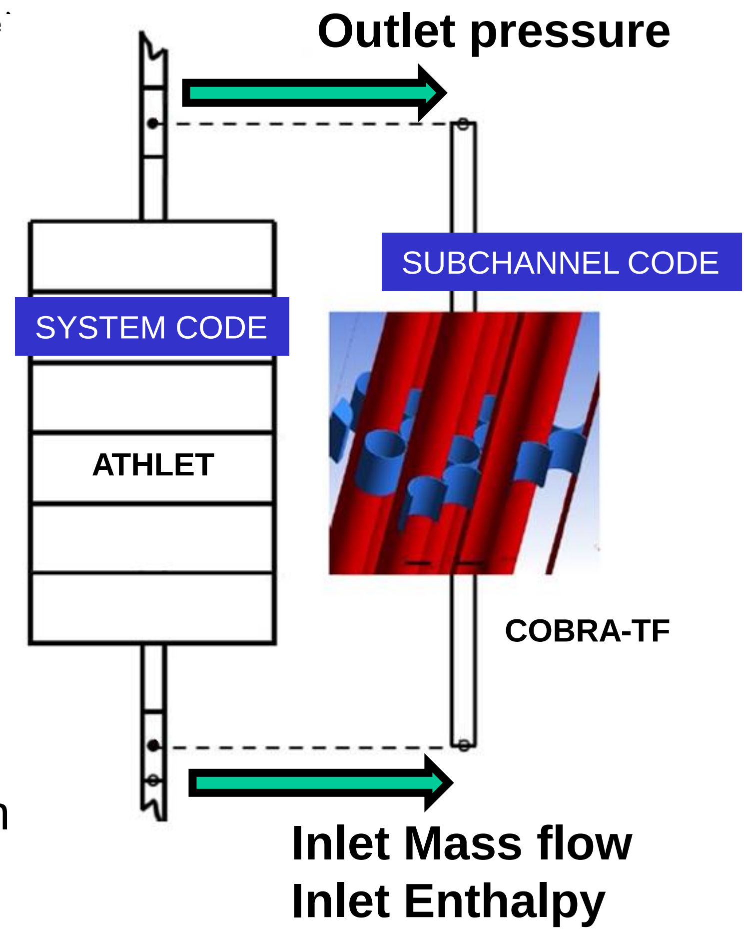
Get initial ATHLET steady state

Coupling script (Part 3) Steady State

```
# Give BC to CTF
massflow=a30.getOutputMEDField("inlet_massflow")
pressure=a30.getOutputMEDField("outlet_pressure")
enthalpy=a30.getOutputMEDField("inlet_enthalpy")
ctf.setInputMEDField("inlet_massflow",massflow)
ctf.setInputMEDField("inlet_enthalpy",enthalpy)
ctf.setInputMEDField("outlet_pressure",pressure)

#Do CTF iteration for Steady state
conv = False
while conv ==  False:
   dt=ctf.computeTimeStep()
   ctf.initTimeStep(dt)
   success=ctf.solveTimeStep()
   if success:
     ctf.validateTimeStep()
   else:
     ctf.abortTimeStep()
   conv=ctf.isStationary()

# Initialize transient calculation in COBRA-TF
tran="yes"
ctf.transientMode()
```



COBRA-TF always perform a transient solution

Coupling script (Part 4) Transient

```
if  typecase=="transient":
   while(status==0):
      # Chose minimum time step size between ATHLET and CTF
      dta=a30.a_GetTimeStep()
      dtc=ctf.computeTimeStep()
      if dta > dtc:
        dt=dtc
      if dtc > dta:
        dt=dta
      a30.a_SetTimeStep(dt)
      ctf.initTimeStep(dt)
      ttime+=dt
      # Perform ATHLET time step
      a30.a_Trans_Loop(0,ttime)
      # Give BC to CTF
      massflow=a30.getOutputMEDField("inlet_massflow")
      pressure=a30.getOutputMEDField("outlet_pressure")
      enthalpy=a30.getOutputMEDField("inlet_enthalpy")
      ctf.setInputMEDField("inlet_massflow",massflow)
      ctf.setInputMEDField("inlet_enthalpy",enthalpy)
      ctf.setInputMEDField("outlet_pressure",pressure)
      # Do CTF time step
      success=ctf.solveTimeStep()
      if (success == False):
          print "Error in CTF transient"
          break
      if success:
        ctf.validateTimeStep()
      if ttime >= Tend : status=1
      told=ttime
```

Set the minimum time step

Advance ATHLET solution

Advance COBRA-TF solution

Testing suite

- **The following cases are available on the NURESAFE repository**
  - ATHLET/DYN3D
    - 5x5 Minicore HFP, Steady-State, Inlet temperature transient
    - 5x5 Minicore HZP, Steady-State, Rod ejection transient
    - 7 FA Hexagonal Minicore, HFP, Steady-State
  - CTF/DYN3D
    - 5x5 Minicore HFP, Steady-State, Inlet temperature transient
    - 5x5 Minicore HZP, Steady-State, Rod ejection transient
    - 7 FA Hexagonal Minicore, HFP, Steady-State
  - ATHLET/CTF(/+DYN3D)
    - 5x5 Minicore HZP, Steady-State, Inlet temperature transient

Open questions ATHLET/CTF

- Most probably the ATHLET API will have to be changed in order to allow different meshing in ATHLET and CTF.

- Check if it is possible to get the power as boundary condition and not from a NK code.

- Both points are being checked at GRS.

Outline

- Short review of work done within WP3.3
- Description of ATHLET SALOME component
- Description of COBRA-TF SALOME component
- Description of the multi-scale coupling ATHLET/COBRA-TF
- **Application to BWR-ATWS**
- Conclusion & Outlook

Scope of the multiscale coupling within WP3.3

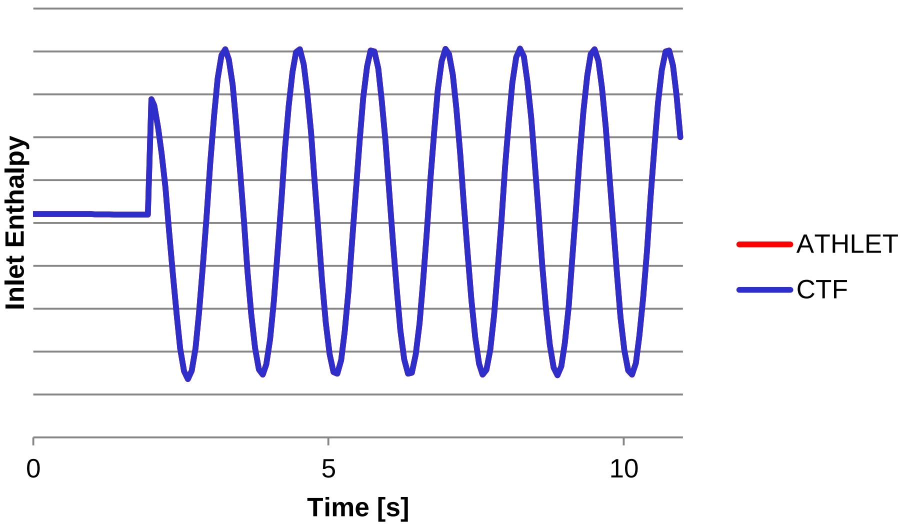
- **ATHLET with a coarse core model and CTF full core assembly-wise**
  - ATHLET using 6 channel model of the O2 core
  - COBRA-TF using 444 channel model of the O2 core

- **ATHLET assembly-wise and one or several bundle with CTF pin-wise**
  - ATHLET 222/444 channel model of the O2 core
  - COBRA-TF at sub-assembly level in one or few FA.

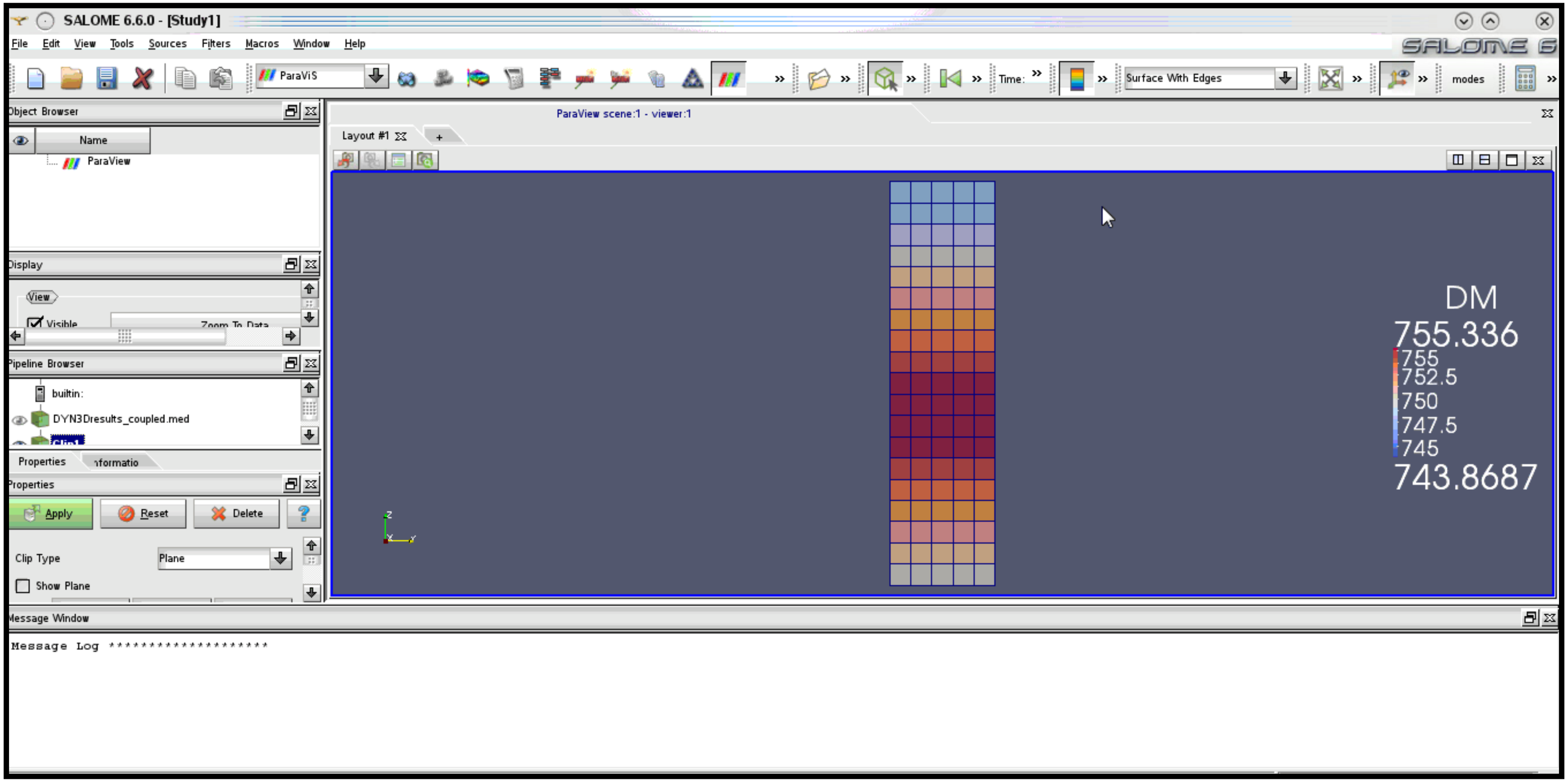
Example test problem

- **ATHLET/CTF recmini25_hzp**
  - Sinusoidal inlet enthalpy perturbation
  - Data is successfully transfered between the codes

	1	2	3	4	5
A	REFL	REFL	REFL	REFL	REFL
B	REFL	MOX 4.3 %	MOX 4.3 %	MOX 4.3 %	REFL
C	REFL	MOX 4.3 %	UO2 4.5 %	MOX 4.3 %	REFL
D	REFL	MOX 4.3 %	MOX 4.3 %	MOX 4.3 %	REFL
E	REFL	REFL	REFL	REFL	REFL



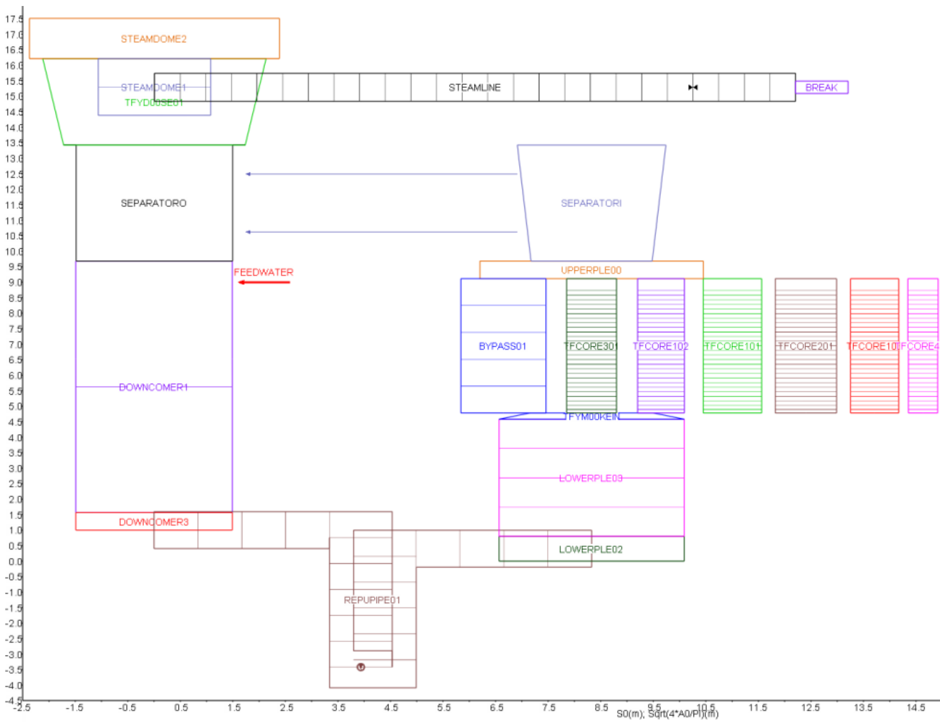
Example test problem



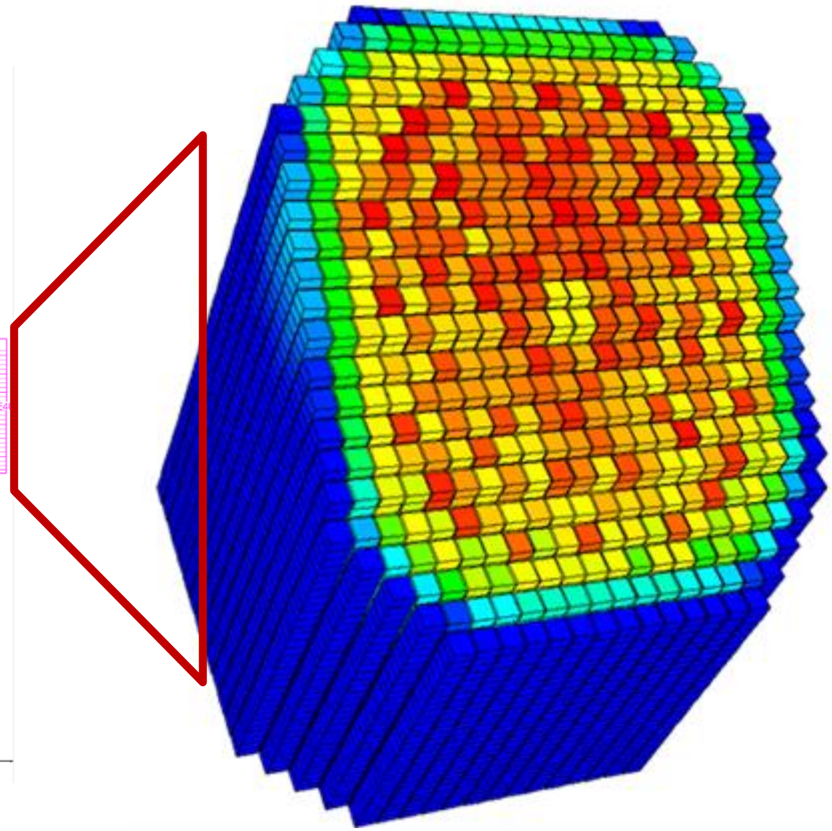
See Movie

ATHLET/COBRA-TF for BWR ATWS

- **FUTURE WORK in the next months:**
  - ATHLET/COBRA-TF multi-scale simulations of the Oskarshamn-2 core.



ATHLET



COBRA-TF

Outline

- Short review of work done within WP3.3
- Description of ATHLET SALOME component
- Description of COBRA-TF SALOME component
- Description of the multi-scale coupling ATHLET/COBRA-TF
- Application to BWR-ATWS
- **Conclusion & Outlook**

Conclusions and Outlook

- **Remaining tasks within WP3.3**
  - Development of a multi-scale simulation approach of a BWR ATWS transient using ATHLET/CTF.
    - All the tools are available.
  - Multi-scale BWR simulations using ATHLET/CTF and comparison with other available transient solutions.
    - This task will continue in the next months, once the activities within WP1.3 are clarified.
- **Good on time for the next report (T0+36)**
  - D33.12.5 Report about multi-scale simulation of a BWR ATWS transient.