



OpenTURNS and HPC with SALOME platform

A. Geay
EDF R&D

HPC and Uncertainty Treatment
Examples with OpenTURNS
EDF – PhiMeca – IMACS -Airbus Group – CEA

Prace Advanced Training Center
May 19th 2016



MAISON DE LA SIMULATION

OUTLINE

1. OpenTURNS use in Salome

1. Presentation of Salome
2. OpenTURNS graphical user interface
3. Using OpenTURNS Graphical User Interface
4. Using YACS to call a solver code defined as a Python script
5. Integration of a code for a usage with OpenTURNS in Salome
6. Distribution and future evolutions

2. Examples of OpenTURNS studies with HPC

Presentation of Salome (1/6)

- **What's Salome ?**

- A middleware providing generic functions for numerical simulations
- An open framework to build domain specific solutions

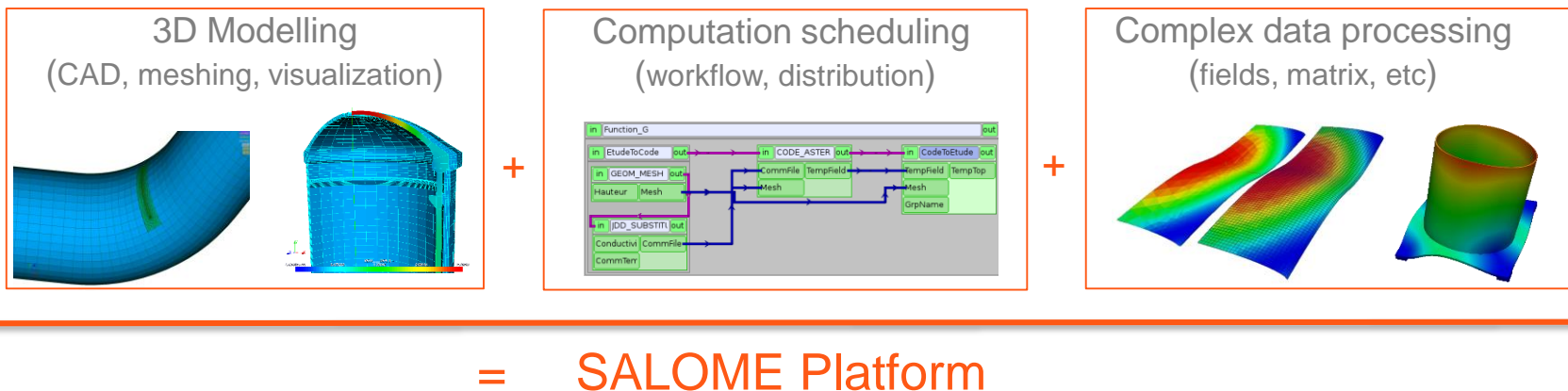
- **Who is developping Salome ?**

- EDF, CEA and OpenCascade partnership



Presentation of Salome (2/6)

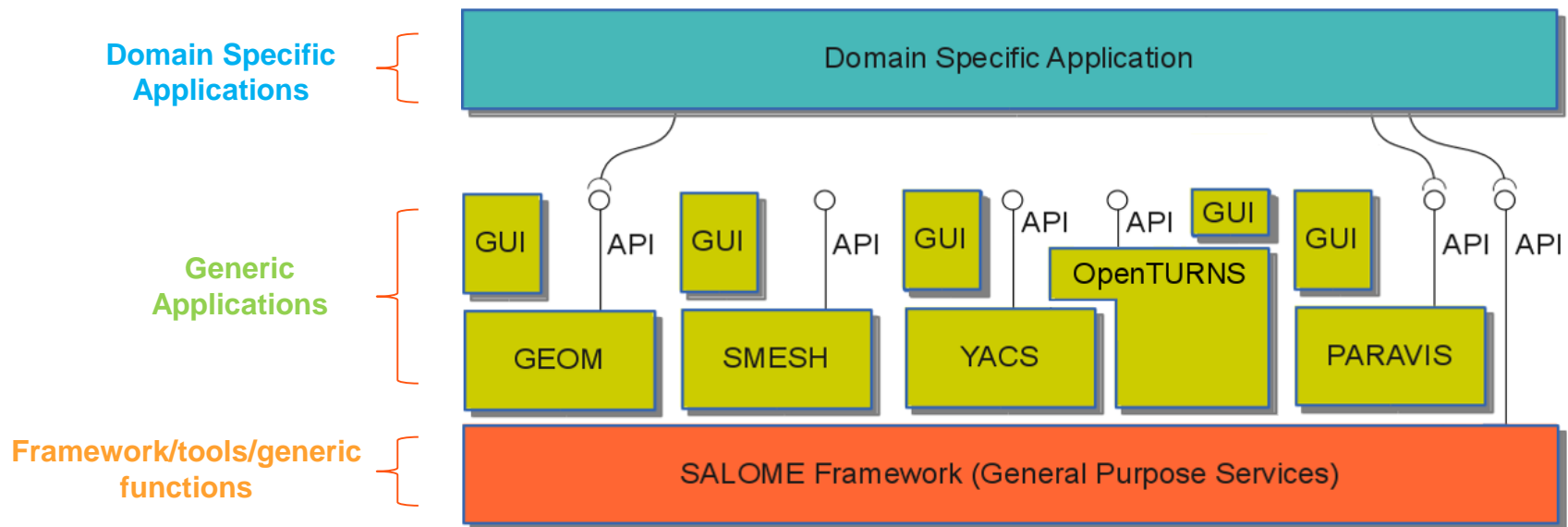
- A middleware providing generic functions for numerical simulations
 - Geometry modelling, meshing, field handling and visualization
 - Data Exchange Model for interoperability of computation codes
 - Computation scheduling (YACS)



Presentation of Salome (3/6)

■ An open framework to build domain specific solutions

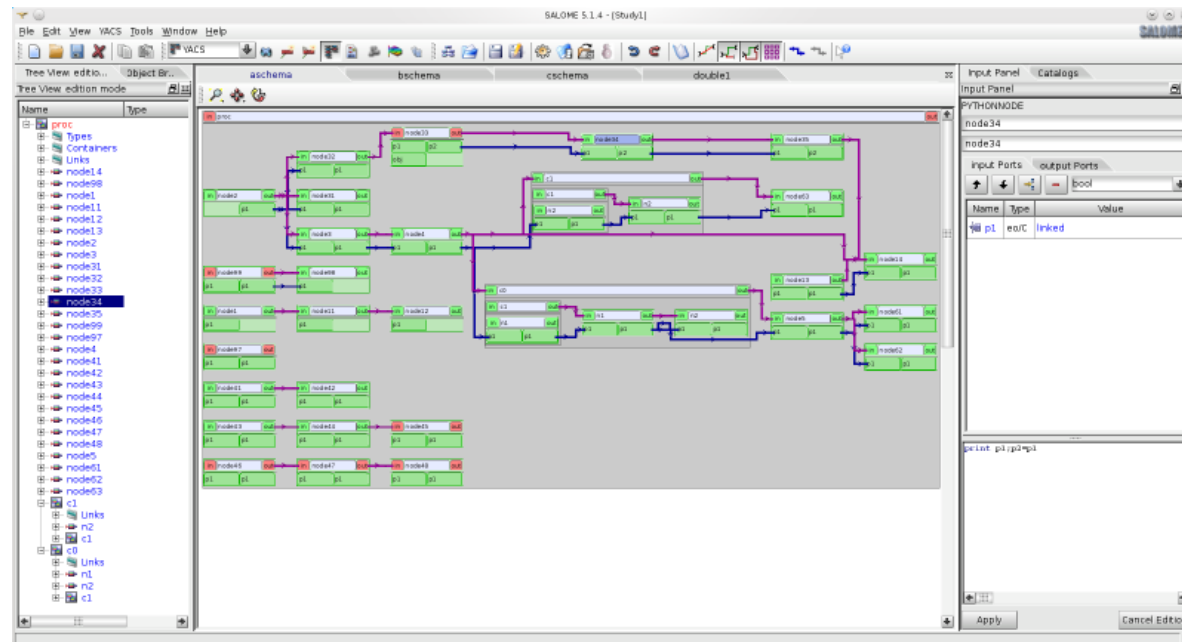
- Each module provides you :
 - A textual interface for scripting based on Python
 - A graphical interface for interactive usages (C++, Qt, PyQt)
 - A programming interface to build custom applications (API C++ and Python)



Presentation of Salome (4/6)

- Presentation of YACS

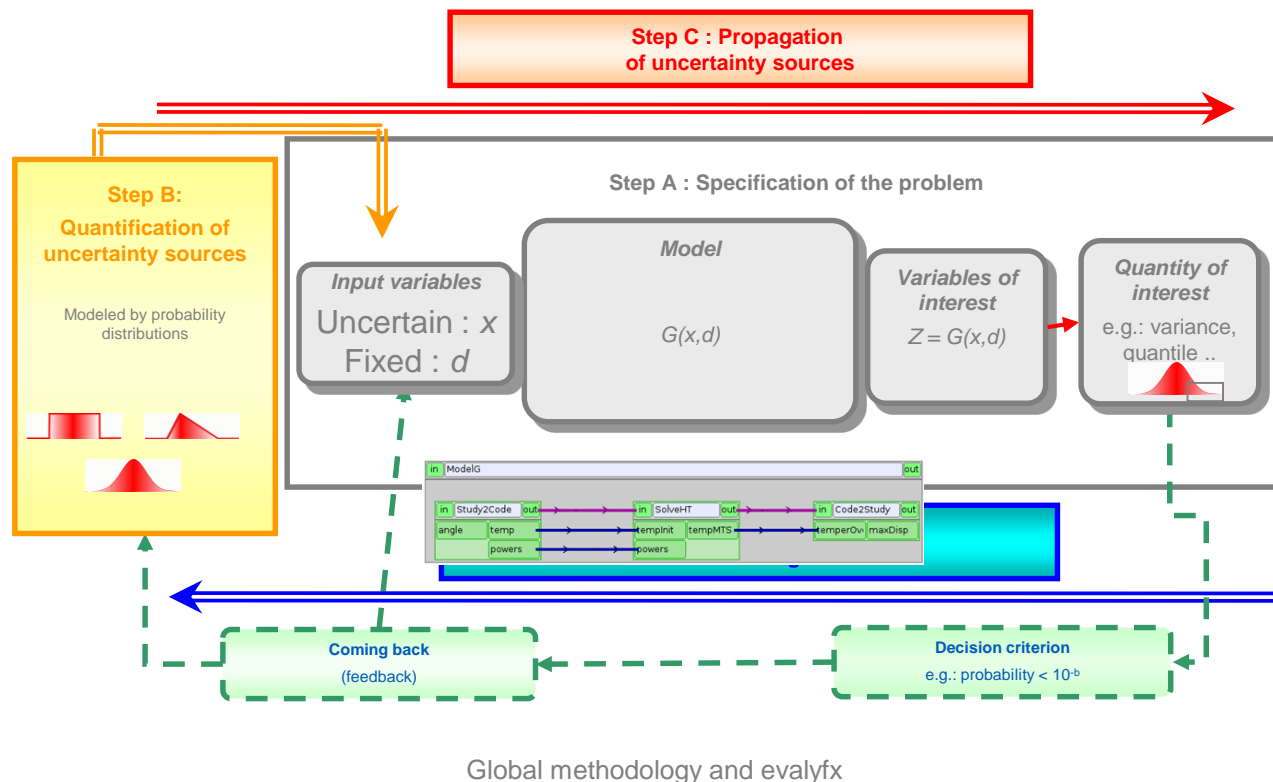
- ▣ Computation supervision module in Salome
- ▣ Define chaining, looping, coupling of computation nodes
- ▣ Define placement constraints of computation nodes
- ▣ Possibility to run python code on different processes (Python interpreter) on a cluster job
- ▣ Supervisor for the execution of computation schema : Possibility to run/stop/step computation, know the progression of the execution of schema, retrieve data during execution



Presentation of Salome (5/6)

■ Presentation of YACS (2)

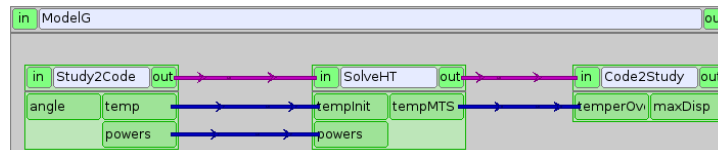
- YACS proposes since 7.7.1 a layer dedicated to uncertainty/parametric studies evaluation called evalyfx.
- Evalyfx has a simple C++/Python API.



Presentation of Salome (6/6)

■ Presentation of YACS (3)

- Evalyfx is dedicated for high computational cost models.
- Evalyfx takes as input a YACS schema containing a model.
- Evalyfx gives access to free inputs and all outputs having type compatible to OpenTURNS.
- User chooses its input variables and variables of interest among returned list above.
- Evalyfx computes complexity of model (level of parallelism) and machines/cluster able to participate to the computation.
- User chooses its machines or cluster
- Evalyfx evaluates by optimizing computing resources



The screenshot shows the OpenTURNS software interface. On the left, the 'OpenTURNS Study tree view' displays a hierarchy: 'OTStudy_0' containing 'physicalModel_0', which further contains 'Deterministic study', 'Probabilistic study', and 'Designs of experiment'. On the right, the 'HTO' window displays the 'Data file' path and a table of 'YACS Scheme Parameters'.

	Name	Description	
1	q		0.2
2	e		0.3
3	c		0.4

	Name	Description	
1	ep		0.9

OpenTURNS Graphical User Interface (1/4)

- Released in SALOME 7.8.0 internal EDF in a couple of weeks 😊
- What's the objective of OpenTURNS GUI ?
 - Guide user in a same homogeneous environment through the roadmap defined by the global methodology of treatment of uncertainties (physical model evaluation and display of result to take decisions...)
- OpenTURNS GUI is more than a simple GUI !
 - OpenTURNS GUI is based on a high level object model (Main study, Deterministic study, Probabilistic study, ...)
 - OpenTURNS GUI includes a layer over OpenTURNS tools.
- How can you use it ?
 - Standalone binary called otgui
 - In a dedicated salome module
 - Can be used in customized salome module

OpenTURNS Graphical User Interface (2/4)

- **Two complementary ways to pilot OpenTURNS GUI : Python and widgets**
 - The design of OpenTURNS GUI allows a strong relationship between Python scripting and graphical interface (Model/View paradigm).
 - Actions you perform on gui can be mapped into a Python representation.
 - Load python script and dump python script.
 - Start session with graphical interface then continue with script then...
 - OpenTURNS GUI offers software bricks usable outside a dedicated tool

OpenTURNS Graphical User Interface (3/4)

- **OpenTURNS GUI is an excellent target for High Performance Computing (HPC)**
 - A large number of independent computations
- **The usage of distributed resources is very dependent on the context:**
 - Use of a cluster (homogeneous, centralized) / a grid (heterogeneous, decentralized)?
 - Communication protocol with the cluster?
 - Which batch / grid manager?
 - Can we install softwares on the cluster?
 - Global / local (by node) filesystem?
 - Execution of OpenTURNS script on the client workstation / on the cluster?
 - Which middleware for the distribution on the cluster?
 - Size of input and output files of the solver code?

OpenTURNS Graphical User Interface (4/4)

- **Which SALOME software bricks are used by OpenTURNS GUI ?**
 - GUI Python console widget
 - YACS engine (3 physical model evaluation channels exist in OpenTURNS GUI. Among them evalyfx)
 - JOBMANAGER engine

- **Why using YACS and JOBMANAGER ?**
 - Computation distribution (YACS supervisor)
 - Launch and monitor remote computations, typically on clusters (JobManager module)
 - Manage the communication with the cluster (SSH or RSH)
 - Manage the file transfer
 - Submission and monitoring of jobs
 - Abstraction of the batch managers (LSF, PBS, Slurm, SGE, LoadLeveler, OAR)
 - Possibility to use different protocols between the nodes (SSH, srun, pbsdsh, ...)
 - Python interpreters over several nodes steering

Using OpenTURNS Graphical User Interface (1/3)

- First step: Definition of the physical model
- Example: Computation of the deviation of a cantilever beam

The screenshot displays the OpenTURNS graphical user interface. On the left, the 'OpenTURNS Study tree view' shows a hierarchy starting with 'OTStudy_0', which contains 'physicalModel_0'. Under 'physicalModel_0', there are four sub-items: 'Deterministic study', 'Probabilistic study', 'Designs of experiment', and 'YACS Scheme Parameters'. The 'physicalModel_0' item is currently selected. The main area on the right, titled 'OT MDI zone', shows the 'Data file' path as '/home/H87074/Documents/OPENTURNS/Beam.xml'. Below this, the 'YACS Scheme Parameters' section is expanded, showing 'Inputs' and 'Outputs' tables.


Inputs

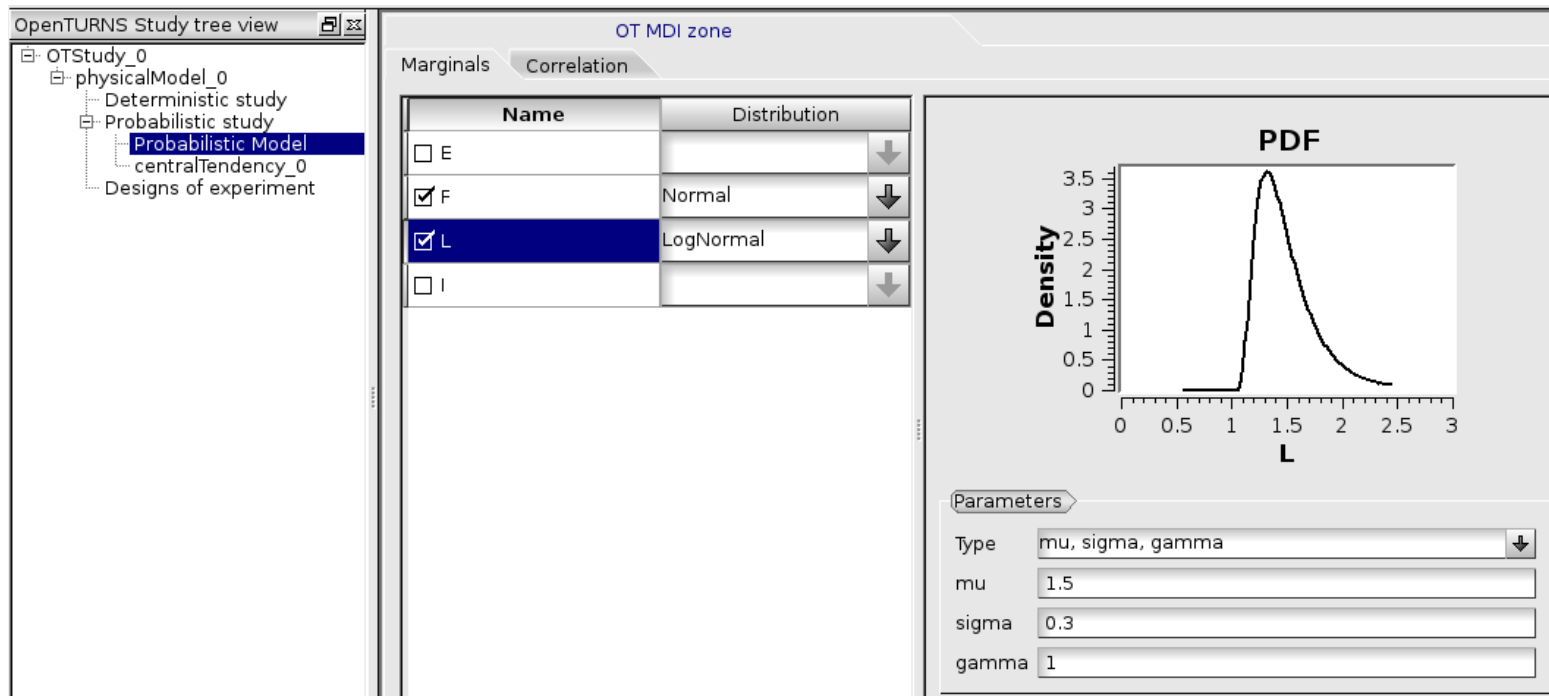
	Name	Description	
1	E		2.1e+11
2	F		1000
3	L		1.5
4	I		2e-06

Outputs

	Name	Description	
1	y		0.00267857

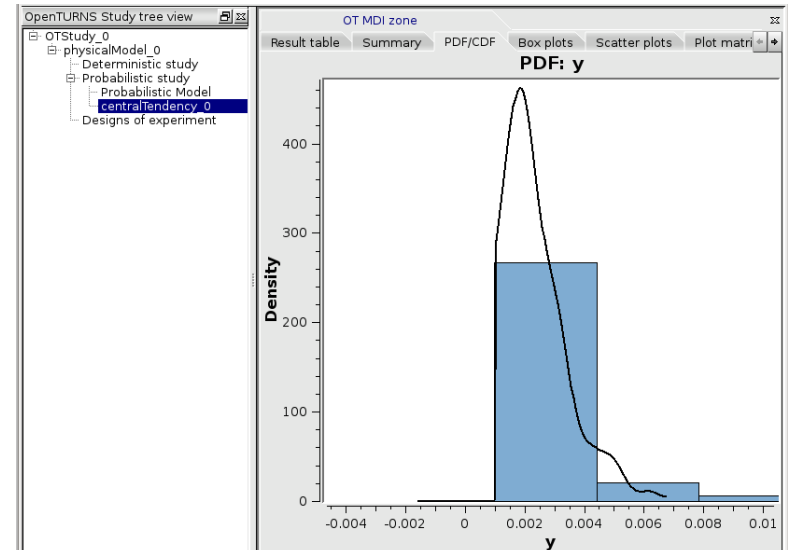
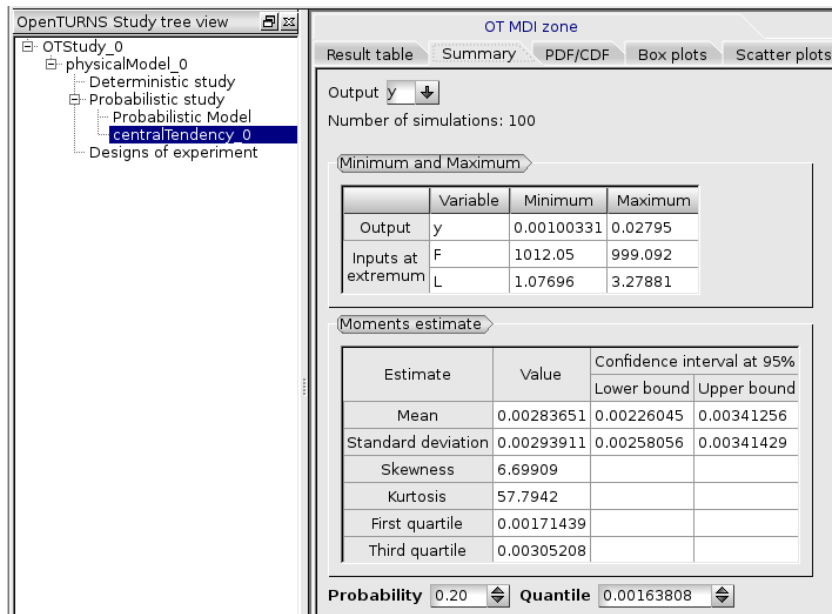
Using OpenTURNS Graphical User Interface (2/3)

- **Third step: Definition of the probabilistic study** 
 - Graphical interface to define a simple study and to check visually the distributions



Using OpenTURNS Graphical User Interface (3/3)

■ Results



Integration of a code for use with OpenTURNS in Salome (1/3)

- There are 3 big categories of code

- Black-Box code

- Code only callable by executing an executable taking files in input (located correctly), a right environment and writing output files, error files...
 - Pros: Statelessness is guaranteed ! Clear separation. No code dependencies. Easy to rerun a failed case. Protected against crash.
 - Cons : Massive usage of files (very poor bandwidth and latency, OS limits of opened files at the same time). Each launch is expensive. Hard to see in details what was wrong in case of failure. Interface is fixed, monolithic, and more limited. The interaction with widgets is less user-friendly.

- Code with python interface :

- Code that offers a python interface.
 - Pros : Exchange of data memory to memory directly. All python friendliness (help context, backtrace in case of error, exceptions, completion). Easiness of interoperability with other python packages (standard or not).
 - Cons : Statelessness is not guaranteed ! Some extra memory copies can occur. No protection against crash.

- Code with a C/C++/FORTRAN high level API.

- Code is incarnated as a library and an associated interface.
 - Pros : Performance can be maximized. No extra cost implied by multi evaluations in a same process.
 - Cons : Statelessness is not guaranteed ! No protection against crash. Laborious debugging.

Integration of a code for use with OpenTURNS in Salome (2/3)

- I've a code with high computational cost. How can I integrate it in YACS to be callable from OpenTURNS GUI ?
 - Black-Box code :
 - Write a wrap python class using standard python modules (like subprocess, tempfile,...). This layer typically : creates a temporary directory, chdir in it, creates inputs files, run code, read output files, clear temporary directory.
 - Use this wrap code in script node in YACS.
 - Code with python interface :
 - Does Python interface exchange pickelizable objects ? If yes, no problem, use it. If no, use standard ones and those available in Salome (fields, meshes, geometries) in a layer.
 - Does Python interface of code allow stateless invocation ? If yes, OK use it directly. If no,..., consider it as a Black-Box code.
 - Code with a C/C++/FORTRAN high level API.
 - Is the code high level interface of code support stateless evaluation ? If yes go ahead, if no, ..., consider it as a Black-Box code.
 - Two possibilities to integrate this type of code :
 - Possibility 1 : Do a python interface based on the C/C++/FORTRAN API.
 - Possibility 2 : Use a tool called YACSGEN working on a FORTRAN/C/C++ API to produce what a SALOME Component.

Integration of a code for use with OpenTURNS in Salome (3/3)

- **Salome provides a tool (YACSGEN) to facilitate the creation of Salome components (for codes presenting a C/C++/FORTRAN API)**
 - See the example script
`SALOME_INSTALL_PATH/appli_V7_7_0/share/salome/resources/genericsolver/yacsgen_example.py`
- **See OpenTURNS module documentation, section « Integration Guide »**
- **Also see YACS module documentation**

Distribution and future evolutions (1/2)

- **Distribution of SALOME + OpenTURNS platform**
 - OpenTURNS and YACS gateways exist since SALOME 5.1.5 (December 2010)
 - but now OpenTURNS GUI is able to call YACS directly without context switching (see slide 5).
 - LGPL license for the whole platform (SALOME + OpenTURNS + OPENTURNS GUI)
 - SALOME platform with OpenTURNS is not distributed yet on <http://www.salome-platform.org>
 - But it is distributed through Salome-Meca (<http://www.code-aster.org/astersalome>)

Distribution and future evolutions (2/2)

■ Developments done recently

- Continue even on error (keepgoing mode)
- Resume after crash or after a user stop
- Improvement of error reporting
- Automatic tuning of YACS graphs to use at most the hardware resources allocated

■ Improvements to come

- Nice GUI for execution follow-up
- Enable the distribution of MPI parallel codes
- Better integration with batch managers
 - Job sizing depending on cluster workload
 - Dynamic adaptation depending on cluster workload
- Integration with Paravis for more visualization possibilities
 - Graphics (Boxplot, etc.)
 - Visualization of fields with uncertainties

OUTLINE

1. OpenTURNS use in Salome

Presentation of Salome

OpenTURNS integration with Salome platform

Documentation

Using OpenTURNS module graphical interface

Using YACS to call a solver code defined as a Python script

Integration of a code for a usage with OpenTURNS in Salome

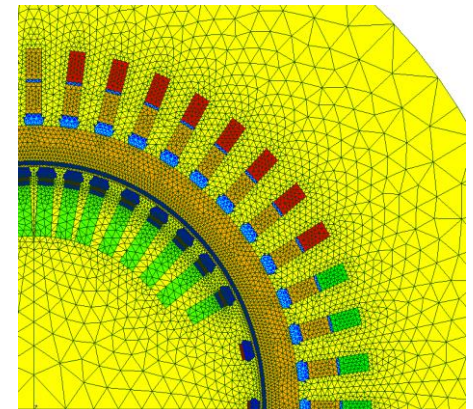
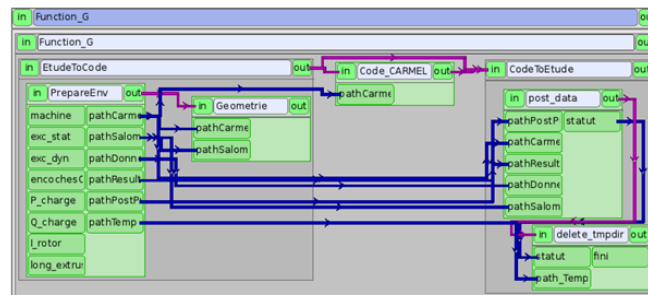
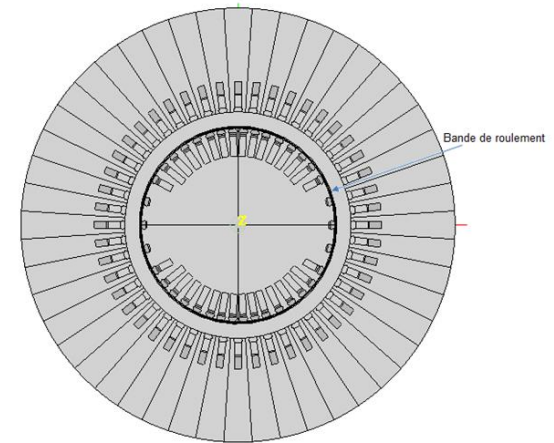
Distribution and future evolutions

2. Examples of OpenTURNS studies with HPC

Examples of OpenTURNS studies with HPC (1/7)

■ Probes in an alternator

- Optimization of the position of probes in an alternator to maximize the signal fidelity
- Solver: CARMEL
- Elementary calculation: Mesh with ~200000 cells. 92 hours
- OpenTURNS study: 115 elementary calculations → about 10000 CPU hours

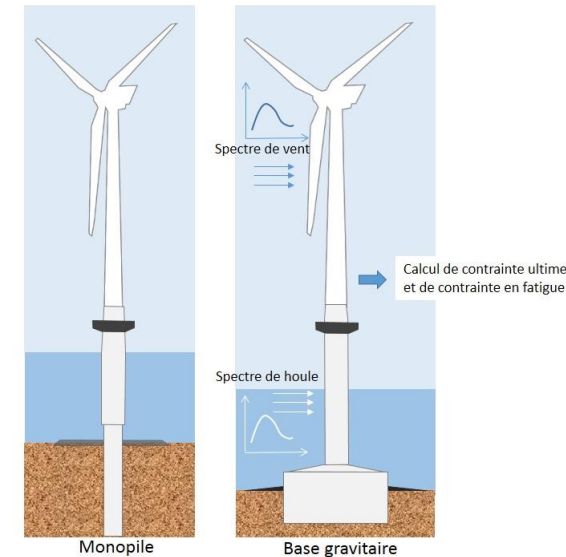


Desquiens 2016

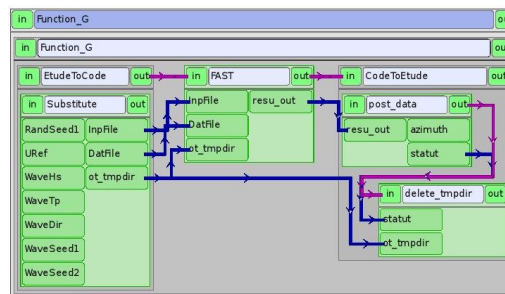
Examples of OpenTURNS studies with HPC (2/7)

■ Offshore wind turbine

- Study of lifetime of offshore wind turbine sustaining extreme scenarios in terms of strain.
- Solver: FAST
- Elementary calculation: No mesh. ~10 minutes.
- OpenTURNS study: 100000 elementary calculations → about 17000 CPU hours



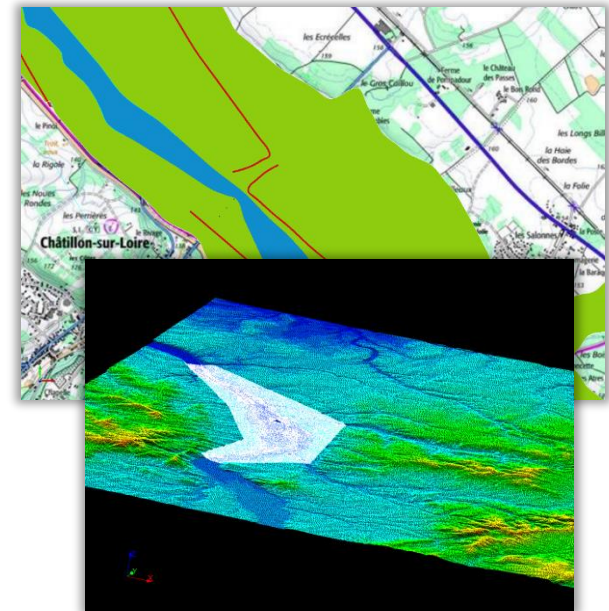
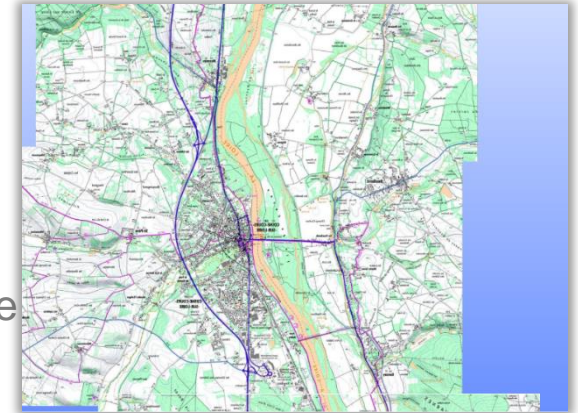
Bousseau 2016



Examples of OpenTURNS studies with HPC (3/7)

- Evaluation of the environment impact of the flood of a river

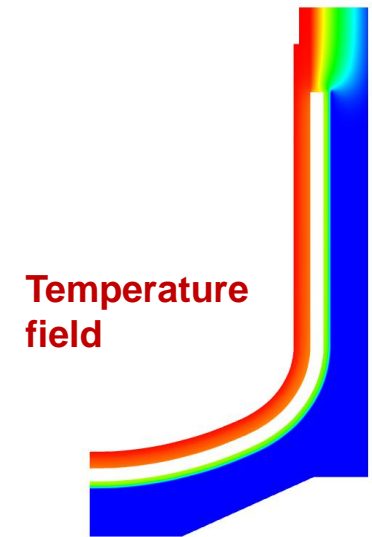
- Solver: TELEMAC/MASCARET
- Elementary calculation: Mesh with ~100000 cells. Convergence reached in about ~1-10 hours
- OpenTURNS study in progress...



Examples of OpenTURNS studies with HPC (4/7)

- **Thermal evacuation of residual power**

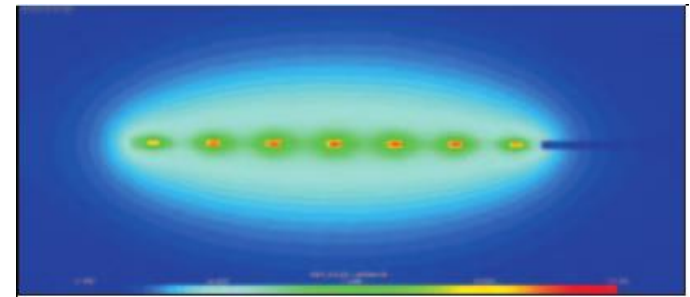
- Uncertainty computation and variable ranking (with 4 uncertain parameters) on the thermal evacuation of residual power through the reactor well in a Gen IV sodium-cooled fast reactor
- Solver: SYRTHES (code for thermal conduction / radiation)
- Elementary calculation: Mesh with 10500 elements. Convergence reached in about 20 seconds on a workstation (one processor only, the elementary case being small)
- OpenTURNS study: 5000 to 10000 elementary calculations → about 50 CPU hours



Examples of OpenTURNS studies with HPC (5/7)

■ HA/LL waste storage

- Uncertainty propagation for the calculation of thermal sizing for the storage of high activity and long life nuclear waste
- Solver: SYRTHES
- Elementary calculation: 10 minutes on 8 cores on a workstation (parallelization in shared memory)
- OpenTURNS calculation: 6000 elementary calculations → about 8000 CPU hours (jobs of 512 elementary calculations on 32 nodes)



Temperature field after several years
in geological storage

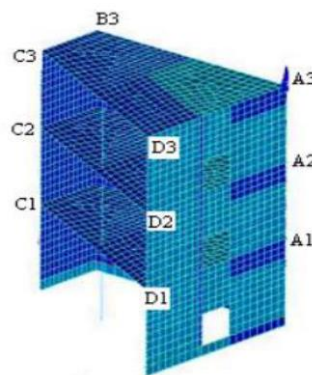
Examples of OpenTURNS studies with HPC (6/7)

■ Seismic resistance with Salome-Meca: SMART Benchmark

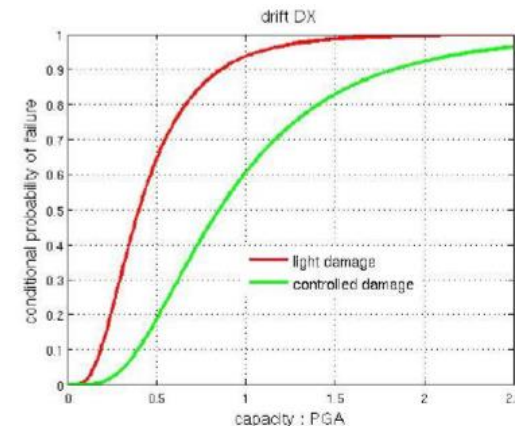
- Fragility curves in the framework of the SMART 2008 benchmark (CEA-EDF)
- FE solver: Code_Aster (structural mechanics)
- Variable of interest: Drift between two levels
 - Two damage categories (3mm and 6mm)
- Uncertainty modelling
 - 50 accelerograms
 - 3 uncertain model parameters



Concrete model used in
SMART benchmark



Mesh used by
Code_Aster



Fragility curves

Examples of OpenTURNS studies with HPC (6/7)

- **Seismic resistance with Salome-Meca: Real application**

- 2012: Reuse of the methodology defined in SMART benchmark for a real industrial structure
- Elementary dynamic computation: 50 to 200h on a workstation
- OpenTURNS study: 186 elementary computations with different parameters and accelerograms
- Computations distributed on 35 CPUs for a total of 425h (~ 18 days) → ~ 15000 CPU hours

Questions?