

# Batch Submission with URANIE

F. Gaudier

[fabrice.gaudier@cea.fr](mailto:fabrice.gaudier@cea.fr)



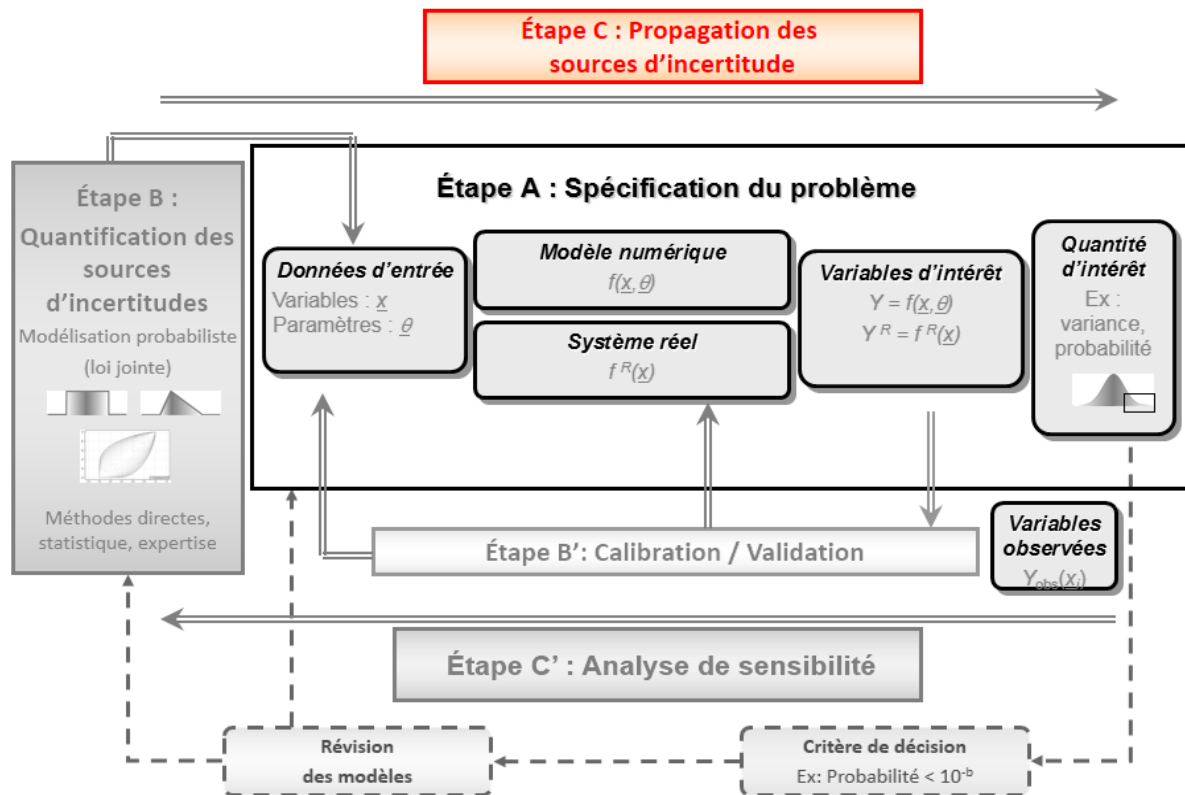
HPC and Uncertainty Treatment  
Examples with OPEN TURNS and URANIE

EDF R&D - Phimeca - IMACS - Airbus Group - CEA

PRACE ADVANCED TRAINING CENTER

Maison de la Simulation, France

2016/05/17-19





# Outline



- External code for the Use Case **Beam**
- Submitting ***Sequential*** job
- Submitting ***Parallel*** job



- A "external code" simulates the "**beam**" problem (if **beam**  $\notin$  \$PATH ; source ...)

```
beam -h
** Usage: ./beam [-v] [-h|-?] [-x [file]]
    -v : Pass to the verbose mode
    -h|-? : Print the usage message
    -x : Set the input file with XML format [beam_input_with_xml.in]
```

- The parameters of the study, with their deterministic values, are:
  - **E** ( $3.0e^7$ ) : the Young modulus
  - **F** (300) : the ponctual load
  - **L** (2.5) : the length of the beam
  - **I** ( $4.0e^{-6}$ ) : the flexion inertia
- The values of these parameters are stored in the "XML" input file "**beam.xml**":

```
<?xml version="1.0"?>
<beam>
<description name="beam" title="UseCase beam with XML input file" version="1.0" date="2014-04-07">
    <tool name="beam exe" version="1.0"/>
</description>
<inputs E="3.0e7" F= "300.0" L="2.5" I="4.0e-6"/>
...
</beam>
```

- Evaluate the external code on the "XML" input file "**beam.xml**" :

```
beam -x beam.xml
```

- The same in "*verbose*" mode (**-v** option)

```
beam -v -x beam.xml
*****
** The Beam use case
** Nb Argument [4]
...
** verbose mode
** XML Input File[beam.xml]
*****
...
*****
** writeXMLOutputs in the XML file [ _beam_outputs_.xml] ...
** End Of writeXMLOutputs with XML file
*****
...
*****
** beam::printLog
** _sFile[beam.xml]
** Inputs : E[3e+07] F[300] L[2.5] I[4e-06] *****
** Output :
** Deviation[13.0208]
...

```

- The target variable is stored in the "XML" output file `_beam_outputs_.xml`

```
<?xml version="1.0"?>
<beam>
<description name="beam" title="UseCase beam with XML input file" version="1.0" date="2014-04-07">
    <tool name="beam exe" version="1.0"/>
</description>
<inputs F="300.0" E="3.0e7" L="2.5" I="4.0e-6"/>
<computation>
    <derivate activate="on"/>
    <hessian activate="off"/>
</computation>
<outputs deviation="1.3020e+01">
    <derivates partialE="-4.340e-07" partialF="4.340e-02" partialL="1.562e+01" partialI="-3.255e+06"/>
    <hessian><partialE partialE="2.893518519e-14" partialF="-1.446759259e-09" partialL="-5.208333333e-07" partialI="1.085069444e-01"/><partialF partialF="0.000000000e+00" partialL="5.208333333e-02" partialI="-1.085069444e+04"/><partialL partialL="1.250000000e+01" partialI="-3.906250000e+06"/><partialI partialI="1.627604167e+12"/></hessian></outputs></beam>
```



- Evaluate a LHS DoE of  $nS = 100$  patterns with the Uranie macro "*macroLauncher.C*"

```
root -l macroLauncher.C
```

- which gives

Processing macroLauncher.C...

...

<URANIE::INFO> \*\* **Proc : 1**

<URANIE::INFO> (100%, time left: 0 sec : Failure 0)

<URANIE::INFO> time elapsed 14.8 sec

<URANIE::INFO> \*\* Clean the Working Directory ... Done

...

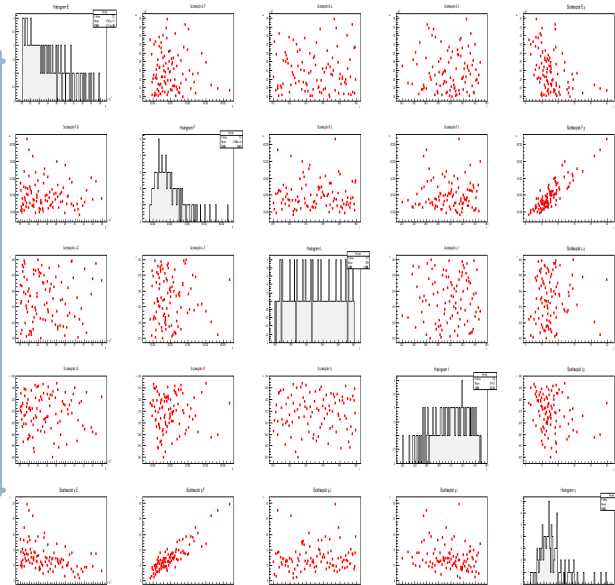
\*\* npatterns[1000]

\*\* y Mean[12.623]

\*\* y Sigma[4.29351]

\*\* L Mean[255]

\*\* L Sigma[2.8881]





- Cluster : **Poincarre** (92 nodes, 2 processors/nodes and 8 cores/processors)
- Use the **LoadLeveler** (IBM) as Job Submission
  - List the tools in your environment

```
module list
```

- List the tools available

```
module avail
```

- Load a tool in your environment

```
module load tool/version
```

- Unload a tool from your environment

```
module unload tool/version
```

# Parallel Job - "LoadLeveler"

- Useful commands of **LoadLeveler**
  - Get the number of free resources

```
llclass
```

- Submit the job file "*batch.sh*" (describe in a next slide)

```
llsubmit batch.sh
```

- List LoadLeveler jobs in the cluster

```
llq <-u user>
```

- Delete a LoadLeveler job

```
llcancel JobID
```

- Help on the Poincare cluster



- Job submission file "*batch.sh*"

```
#!/bin/bash
#@ job_name           = BeamLauncher      # Request name
#@ total_tasks        = 32                 # Number of tasks to use
#@ node               = 2
#@ node_usage         = not_shared
#@ wall_clock_limit   = 00:30:00          # Elapsed time limit in seconds of the job (default: 1800 = .5h)
#@ output             = run_$(jobid).o    # Standard output with the job id
#@ error              = run_$(jobid).e    # Error output with the job id
#@ class              = clallmnds         # Choosing standard nodes
#@ job_type           = mpich
#@ queue

# Load the modules
module load intel intelmpi

# Source the environment variables
source /gpfslocal/pub/training/uncertainty_may2015/URANIE/URANIE/3.6.1/uranie.bashrc

# Remove all old files
rm -f run_*.o run_*.e

# Execute the Uranie macro with the -q option to quit ROOT
root -l -q macroLauncher.C
```



- Command to submit the job file "*batch.sh*"

```
llsubmit batch.sh
```



- It is the same macro to launch in the *sequential* mode than in the *parallel* mode
- Operational for other *Job Submission* systems as SGE, LSF and SLURM (*Curie/TGCC*)
- Don't forget the **-q** option in the ROOT command to quit ROOT when the computation is finished and free the core
- Exists also *Parallel* computing in a personal desktop with several cores:

```
launcher->run("localhost=5");
```