Training Session Uranie

Fabrice Gaudier, Nicolas Gilardi, Jean-Marc Martinez, François Bachoc

CEA/DEN/DANS/DM2S/SFME/LGLS

fabrice.gaudier@cea.fr
nicolas.gilardi@cea.fr
jean-marc.martinez@cea.fr
francois.bachoc@cea.fr

Uncertainties Summer School 2011 EDF/CEA/INRIA

INSTN - Cadarache

July 7, 2011

Overview

- Introduction of the ROOT *framework*
- Training the uncertainties platform URANIE
- Flica IV use case

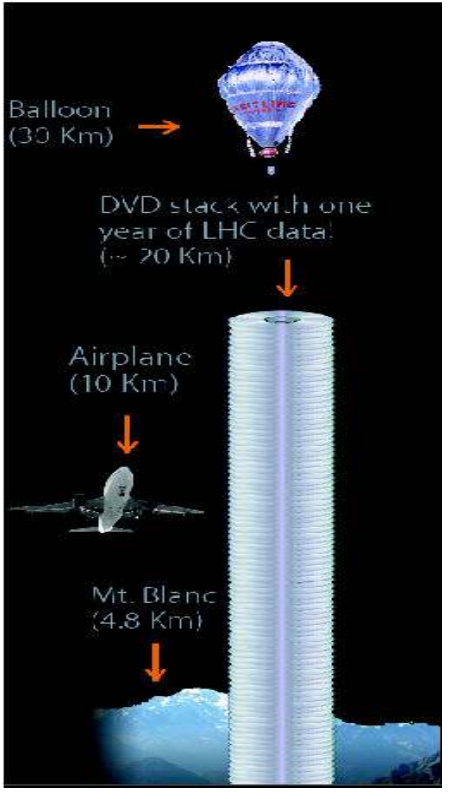
CERN Large Hadron Collider (LHC)

- particle accelerator
- 27 km circumference tunnel in Geneva
- 4 experiments (ATLAS, CMS, ALICE, LHCb)

  Study the structure of matter

  – Search for the Higgs boson

  – Search for new physics
- Data quantity generated : 20 PetaBytes/year
- ROOT is the framework to store, treat and analyze this data

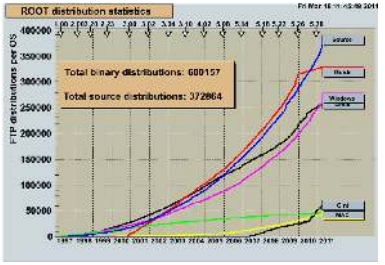


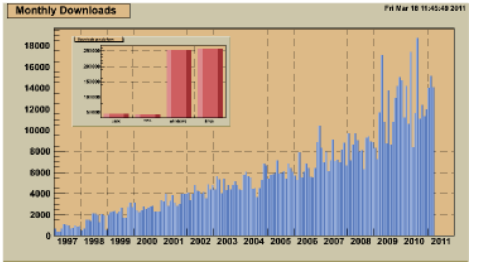
Fons Rademakers / CERN

ROOT (http://root.cern.ch)

ROOT is an object-oriented framework for large scale data analysis and data mining.

- 20 years of development (C++ with 3-4 releases/year)
- multi-plateform (Unix, Windows, Mac OS X)
- Offer :
  - A C++ interpreter, but also python (PyROOT), ruby
  - A hierarchical object-oriented database (machine independent, highly compressed, supporting schema evolution and object versioning)
  - Shared librairies (*automatic loading with "rootmap"*)
  - Advanced statistical analysis tools (subprojects *RooStats*, *RooFit*, *TMVA*)
  - Advanced visualization tools
- **LGPL License**

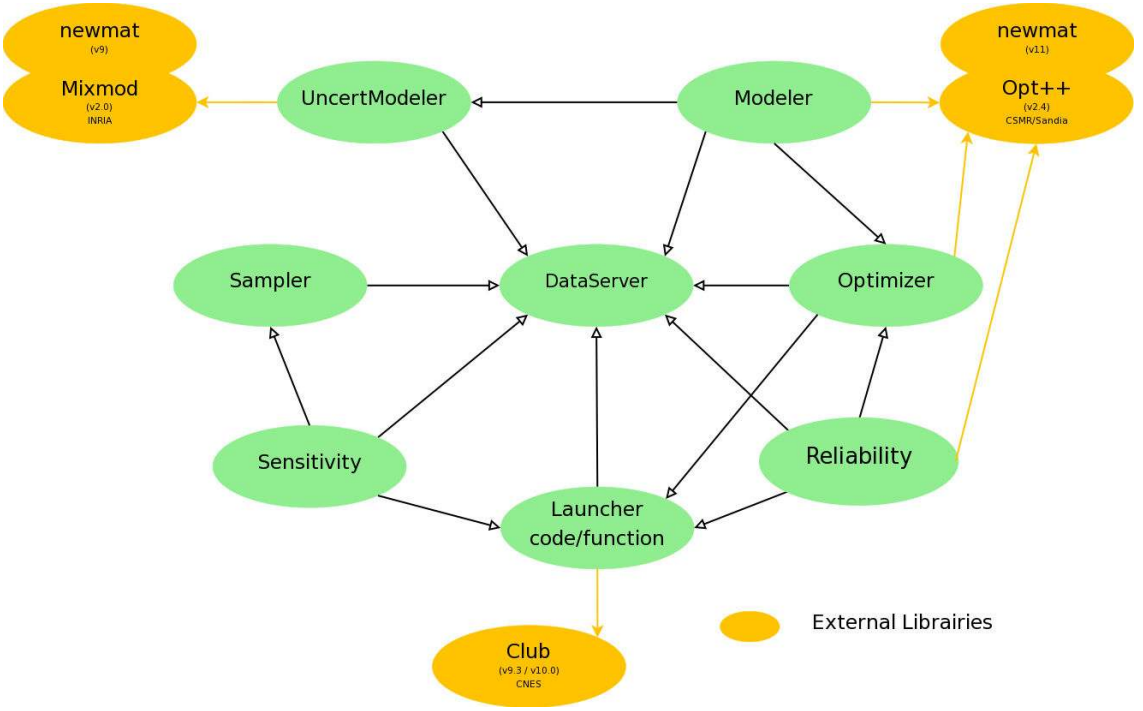




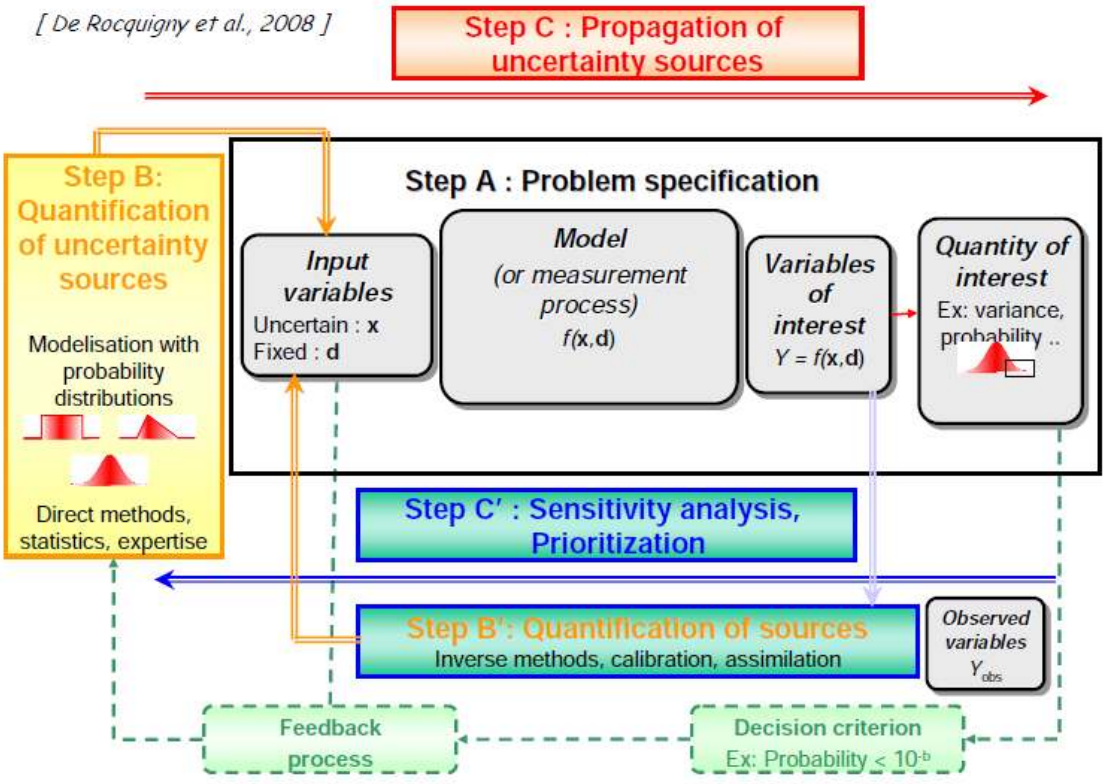
URANIE : CEA/DEN Uncertainty Platform

-  ROOT (CERN),  MIXMOD (Gaussian Mixtures - INRIA),  OPT++ (Optimization - Sandia), CLUB (Text parsing - CNES)
- Data access :
  - Flat file with header ("Salomé Table")
  - TTree (internal ROOT)
  - SQL Data base (MySQL, PostgreSQL, ...)
- Uncertainty/Sensitivity methods in URANIE
  - Design Of Experiments (SRS, LHS, ROA, qMC, MCMC, Copulas)
  - Clustering methods
  - Surrogate models (Polynomial, Artificial Neural Networks, Splines)
  - Non Intrusive Spectrale Projection : Generalized Polynomial Chaos
  - Sensitivity Analysis (Pearson, Spearmann, Morris, Sobol, FAST & RBD)
  - Optimization, Multi-Criteria (**Vizir** library : Genetic Algorithms)
  - Computing distribution

URANIE : Fonctional diagram



Uncertainties overview



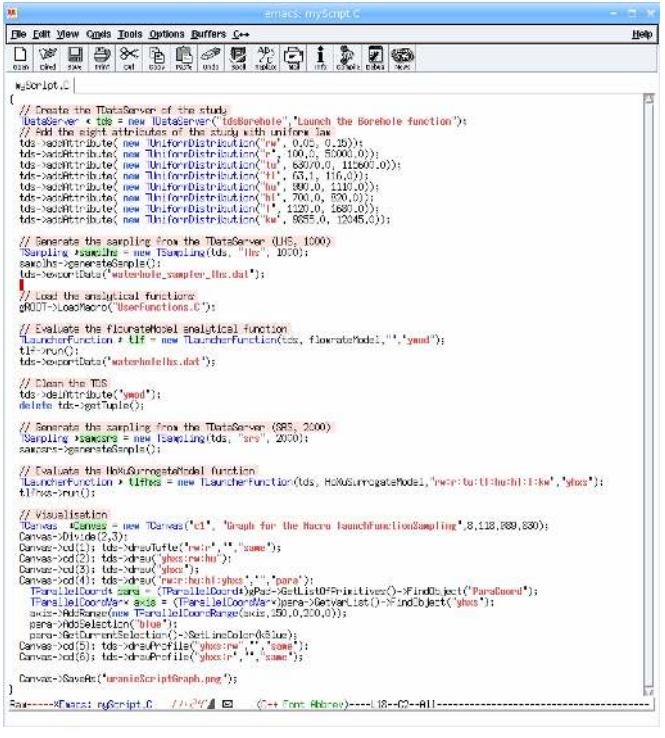
URANIE : Size

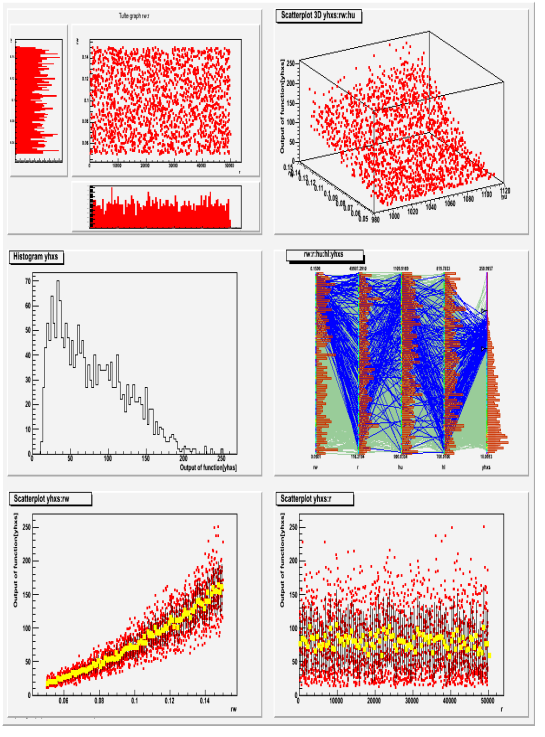
version : v2.3.0
date : 2011/01/14

Libraries	Lines (*.h, *.cxx)	Classes
DataServer	19 000	35
Launcher	9 000	20
Sampler	11 000	22
Modeler	11 000	9
Optimizer	5 000	11
Sensitivity	5 000	9
UncertModeler	2 000	5
Total	**62 000**	**111**

URANIE : Batch mode

> root myScript.C





URANIE - XML User Interface

```
 1<?xml version="1.0" encoding="iso-8859-1"?>
 2<!DOCTYPE Problem SYSTEM "/uranie.dtd">
 3<Problem>
 4  <Header name="boreholeXML" title="Launch the Borehole function in XML" debug="0">
 5    <Application name="uranie" version="1.0"/>
 6  </Header>
 7  <DataDictionary>
 8    <DataField name="rw" law="uniform" min="0.05" max="0.15"/>
 9    <DataField name="r"  law="uniform" min="100.0" max="50000.0"/>
10    <DataField name="tu" law="uniform" min="63070.0" max="115600.0"/>
11    <DataField name="tl" law="uniform" min="63.1" max="116.0"/>
12    <DataField name="hu" law="uniform" min="990.0" max="1110.0"/>
13    <DataField name="hl" law="uniform" min="700.0" max="820.0"/>
14    <DataField name="l"  law="normal" mean="1120.0" std="12.25"/>
15    <DataField name="kw" law="uniform" min="9855.0" max="12045.0"/>
16  </DataDictionary>
17  <Sampler method="LHS" N="1000" export="waterhole_sampler_lhs.dat"/>
18  <Launcher macro="UserFunctions.C" function="flowrateModel" output="ymod" export="waterholelhs.dat"/>
19  <Sampler method="SRS" N="2000"/>
20  <Launcher function="HoXuSurrogateModel" input="rw:r:tu:tl:hu:hl:l:kw" output="yhxs"/>
21</Problem>
```

```
void evaluateXMLFile (TString xmlFile = "uranieproblem.xml")
{
TXMLProblem * xmlProblem = new TXMLProblem(xmlFile);
xmlProblem->submit();
}
```

Projects using URANIE

- LEONAR tool for severe accidents in french nuclear reactor (**CEA-EDF**)
- PSI-Matador Methodology : Dosimetry computation in french nuclear reactor (**CEA-EDF**)
- EHPOC project : Meteor code (**CEA**)
- Sensitivity Analysis for Cathare code (**Areva TA**)
- Multi-criteria optimization (**CEA CESTA/CELIA**)
- Optimization Design for ESS

  The *"European Spallation Source"* is a project to design and construct (Lund - Sweden) a next generation facility for research with neutrons.
- ALLIANCES platform (**CEA/ANDRA/EDF**)

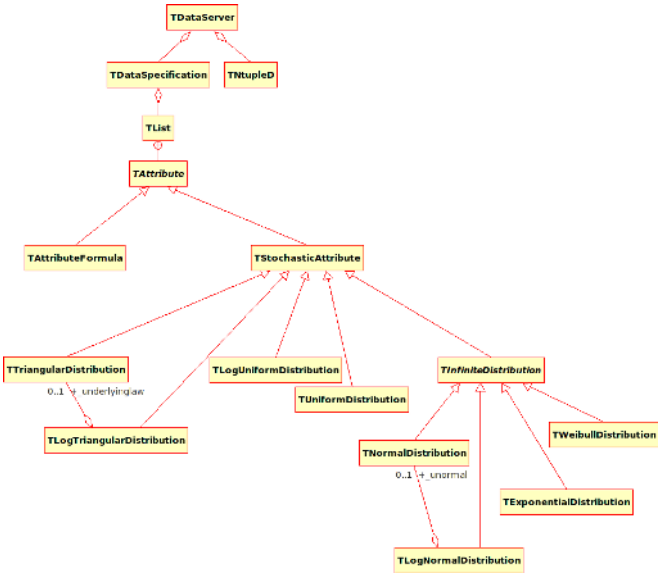
  is to provide a working environment for the simulation and analysis of phenomena to be taken into account for waste storage and disposal studies.
- European project **NURESIM/NURISP**

  The European Platform for NUclear REactor SIMulations, NURESIM, is a Common European Standard Software Platform for modeling, recording, and recovering computer data for nuclear reactors simulations.

"DataServer" library

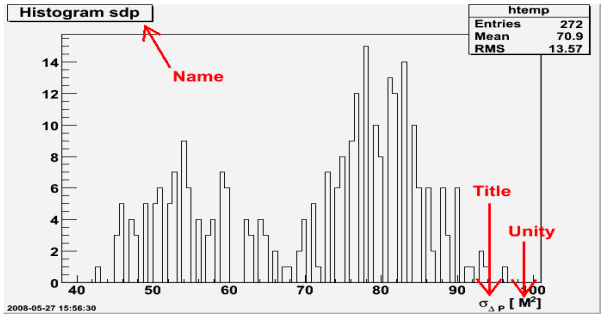
Management of the attributes (∼ variables)
- create/transform attributes
- Load data from external files/formats (Ascii,TTree,Sql)

Graphs and treatments specific to uncertainties



"DataServer" library - URANIE ASCII file format

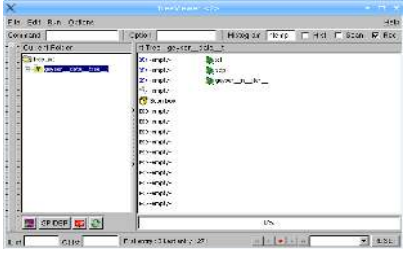
```
#NAME: geyser
#TITLE: geyser data
#DATE: Mon Mar 12 23:41:09 2007
#COLUMN_NAMES: x1| sdp
#COLUMN_TITLES: x_1| #sigma_{#Delta P}
#COLUMN_UNITS: Sec| M^{2}
------- empty line -------
3.600 79.000
1.800 54.000
...
```



tds->draw("sdp");

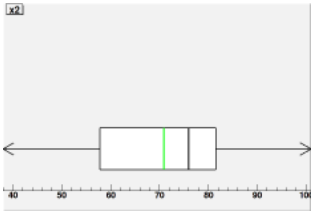
Only the "#COLUMN_NAMES:" line is mandatory
WARNING : the empty line between the header and the matrix data

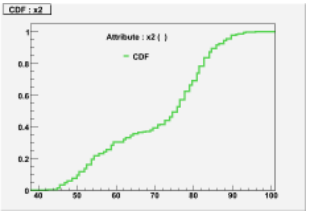
```
TDataServer *tds = new TDataServer();
tds->fileDataRead("geyser.dat");
tds->draw("sdp");
tds->startViewer();
```

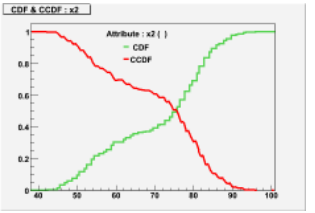


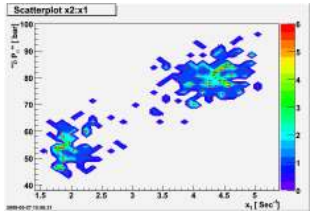
"DataServer" library - statistical graphs

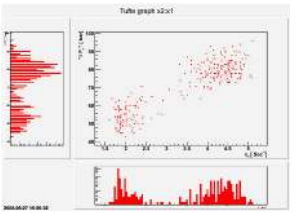
```
tds->drawBoxPlot("x2");
tds->drawCDF("x2","x1<3.0");
tds->drawCDF("x2","x1<3.0","ccdf");
tds->drawScatterplot("x2:x1");
tds->drawTufte("x2:x1");
tds->drawProfile("x2:x1","","same");
```

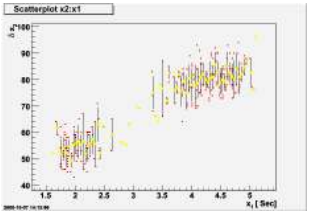




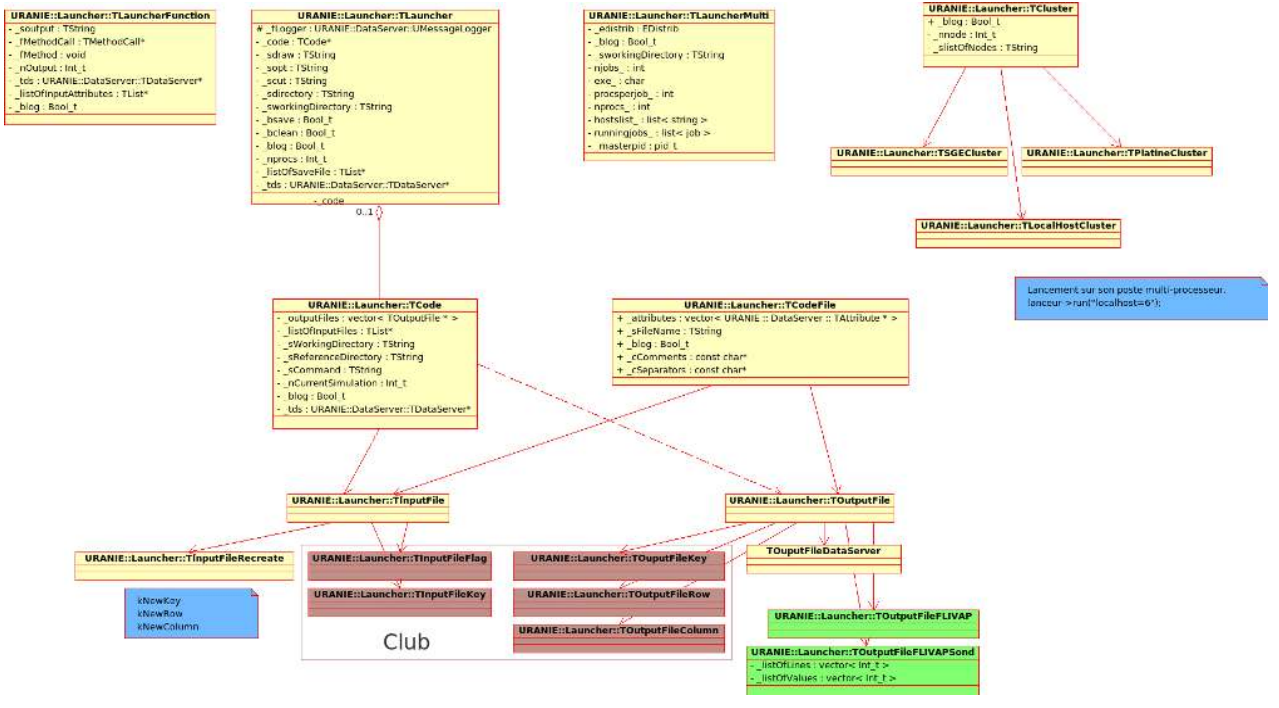








Distribute the model evaluations (sequential, cluster) for:
- Analytical function
- External code



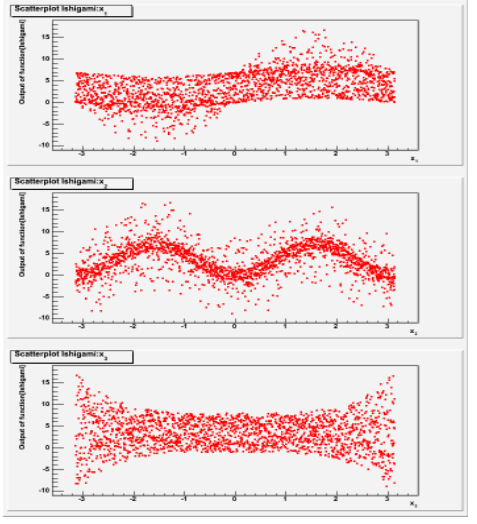
Analytical Function

void myFunction (double *x, double *y)

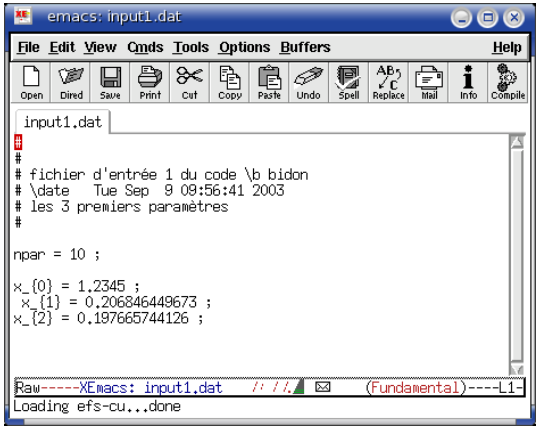
ISHIGAMI benchmark : Analytical function of $R^3 \rightarrow R$ ($x_{i=1,2,3} \sim Unif[-\pi, \pi]$)

$$y = \sin x_1 + A \sin^2 x_2 + B x_3^4 \sin x_1 \qquad A = 7, B = 0.1$$

```
#include "TMath.h"

void Ishigami(double *x, Double_t *y)
{
  Double_t A = 7.0, B = 0.1;

  Double_t x1 = x[0];
  Double_t x2 = x[1];
  Double_t x3 = x[2];

  y[0]  = Sin(x1) + A * Sin(x2) * Sin(x2) + B * Power(x3, 4.) * Sin(x1);
}

{
  TDataServer * tds = new TDataServer("tds", "TDS for Ishigami function");
  tds->addAttribute ( new TUniformDistribution("x_{1}", -1.* Pi(), Pi()));
  tds->addAttribute ( new TUniformDistribution("x_{2}", -1.* Pi(), Pi()));
  tds->addAttribute ( new TUniformDistribution("x_{3}", -1.* Pi(), Pi()));

  TSampling *sampling = new TSampling(tds, "lhs", 1000);
  sampling->generateSample();

  TLauncherFunction * tlf = new TLauncherFunction(tds, Ishigami);
  tlf->run();
}
```



External code : Input Files with " **Key - Value**" 1/3

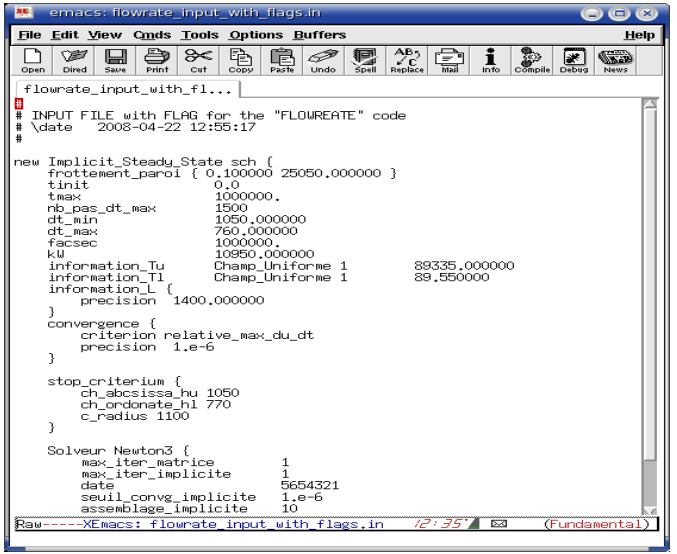



```
TAttribute *x1 = new TAttribute("x_{1}", 0.20, 0.40);
x1->setFileKey("input1.dat");

TAttribute *x2 = new TAttribute("x2", 0.15, 0.25);
x2->setFileKey("input1.dat", "x_{2}");
```

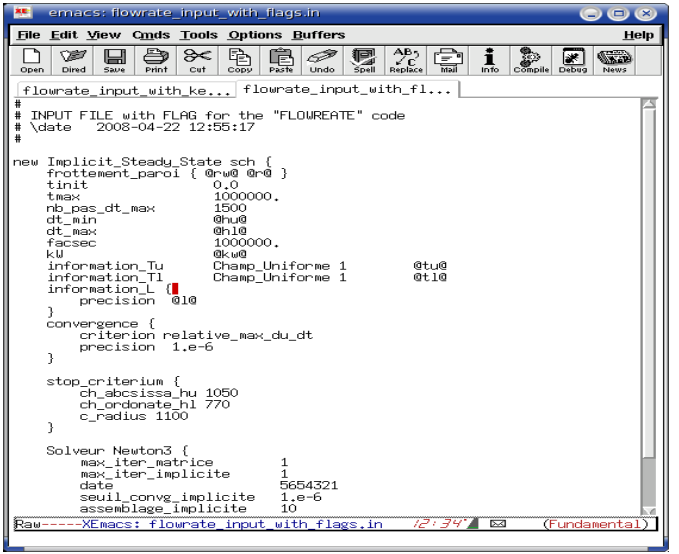
Hypothesis : unicity of the key

```
void TAttribute::setFileKey(
    TString sfile,
    TString skey="",
    TString sformatToSubstitute="%e",
    TAttributeFileKey::EFileType FileType=TAttributeFileKey::kKey);
```

External code : Input Files with "flag" 2/3



Original file



User Flag file

```
attrw->setFileFlag("myfile.in", "@tu@");
attrw->setFileKey("myfile.in", "@tu@", "%f",TAttributeFileKey::kFlag);
```

Hypothesis : unicity of the key not required but intervention of the user

External code : Input Files Create On The Fly 3/3

- **TAttributeFileKey::kNewRow** 4^{th} argument in the **setFileKey** method

```
2.481733e+02 6.112975e-03 1.055352e-06 2.635758e-03 2.217372e+02 1.888999e+00 ...
```

- **TAttributeFileKey::kNewColumn** 4^{th} argument in the **setFileKey** method

```
2.481733e+02
6.112975e-03
...
```

- **TAttributeFileKey::kNewKey** 4^{th} argument in the **setFileKey** method

```
t = 2.481733e+02 ;
kl = 6.112975e-03 ;
kc = 1.055352e-06 ;
...
```

Hypothesis : The input files does not exist

External code : Output Files formats 1/2

- **TOutputFileRow** class

```
0.20E+05 0.5579978E-25 0.2789989E-25
0.30E+05 0.5121863E-20 0.2560931E-20
0.40E+05 0.8212720E-17 0.4106360E-17
0.50E+05 0.1432418E-14 0.7162090E-15
...
```

- **TOutputFileColumn** class

```
0.20E+05 0.30E+05 0.40E+05 0.50E+05 ....
0.5579978E-25 0.5121863E-20 0.8212720E-17 0.1432418E-14 ...
0.2789989E-25 0.2560931E-20 0.4106360E-17 0.7162090E-15 ...
...
```

External code : Output Files formats 2/2

- **TOutputFileKey** class

```
yhat = 3.591931e+01;
d = 2.415401e+03;
...
```

- **TOutputFileDataServer** class

```
#COLUMN_NAMES: yhat | d

3.591931e+01 2.415401e+03
...
```

Distribution 1/2

- PC multicores

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

- Cluster (LSF, SGE)

```
#BSUB -n 10
#BSUB -J FlowrateSampling
#BSUB -o FlowrateSampling.out
#BSUB -e FlowrateSampling.err
source /home/cont002/uranie/uranie-titane.cshrc
rm -f FlowrateSampling.out FlowrateSampling.err
root -l -q lanceurFLOWRATE_SAMPLING.C
```

> bsub < BsubFile

Distribution 2/2

- Salomé/Uranie : Share the new Salomé launcher in Uranie

```
<?xml version="1.0" encoding="iso-8859-1"?>
<main>
 <machine-list>
   <machine env-file="/home/cont002/uranie/uranie-titane.cshrc"
      work-directory="/work/cont002/gaudier/testKERNELSALOME_titane">titane
 </machine>
   <machine env-file="/home/uranie/uranie.cshrc"
      work-directory="/work/gaudier/testKERNELSALOME_awa">awa</machine>
 </machine-list>
 <ref-directory>/home/gaudier/tmp/testuranie/testKERNELSALOME</ref-directory>
 <nb-processes>64</nb-processes>
 <input-file>lanceurFLOWRATE_SAMPLING.C</input-file>
 <input-file>flowrate_input_with_keys.in</input-file>
 <input-file>flowrateborhole.dat</input-file>
 <output-file>_flowrate_sampler_launcher_.dat</output-file>
 <command>
  rm -f flowrateSalome.error.log
  rm -f flowrateSalome.output.log
  root -b -l -q lanceurFLOWRATE_SAMPLING.C</command>
</main>
```

> uranieDistrib flowrateSalome.xml titane

"Modeler" library

Create an analytical function ("surrogate model") between Y and X

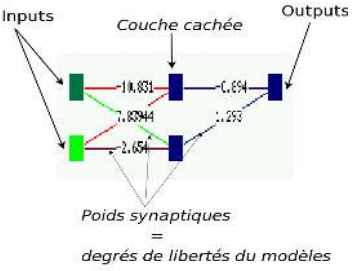
- Learning :
  - ▷ **Opt++** : Levenberg-Marquardt, ...
  - ▷ resampling method : *Bootstrap*, *Leave-one-out*
- Take into account constraints :
  - ▷ Weight sharing

$$\omega_{ij} = \omega_{kl}$$

  - ▷ Physical informations

$$\frac{\partial y_j}{\partial x_i} < 0, \ \frac{\partial^2 y_j}{\partial x_i \partial x_k} > 0, \cdots$$

- Export function in C, C++, Fortran, **PMML** ("*Predictive Model Markup Language*" - **Data Mining Group**)
  - ▷ using for code calibration, propagation of uncertainties, $\cdots$



"Optimizer" library (1/3)

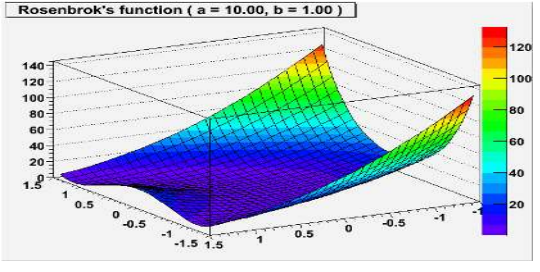
Find the minimun value of a *function*
Identify the parameters of a model from a database

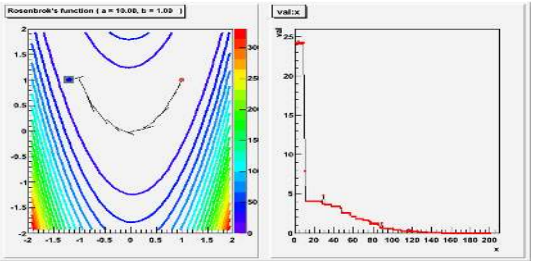
- library **Minuit2** included in **ROOT**
  - Prototype of analytical function :

    void myFunction (Double_t *x, Double_t *y)
  - an URANIE::Launcher::TCode object

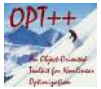
The **Rosenbrock** *benchmark* : $f(x,y) = a(y-x^2)^2 + b(1-x)^2$ with $a = 10.$, $b = 1.$

```
TOptimizer * topt = new TOptimizer(tdsRosenbrock, myRosenbrockCode);
topt->optimize();
```





"Optimizer" library (2/3)



OPT++ is a library of nonlinear optimization algorithms

http://csmr.ca.sandia.gov/opt++/

- Direct, with or without Gradient, Newton, ...
- DLL mode with the Opt++ prototype :

```
void FCN0(int n,const ColumnVector& x,real& fx,int& ret)
void FCN1(int mode,int n,const ColumnVector& x,real& fx, ColumnVector& gx, int& ret)
void FCN2(int mode,int n,const ColumnVector& x,real& fx, ColumnVector& gx, SymmetricMatrix& Hx, int& ret)
```

Example:

```
TDataServer * tds = new TDataServer();
tds->addAttribute( new TAttribute("x1", 2.0, 4.0));
...
TOptimizerOpt *topths65 = new TOptimizerOpt(tds, "hs65.so", "hs65_2", "init_hs65");
topths65->addConstraint("ineq_hs65");
topths65->setFcnTol(1.0e-06);
...
topths65->optimize("nips");
```

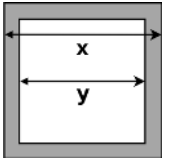
"Optimizer" library (3/3)

Multi-Criteria Optimization by Genetic Algorihtms : CEA library **Vizir**

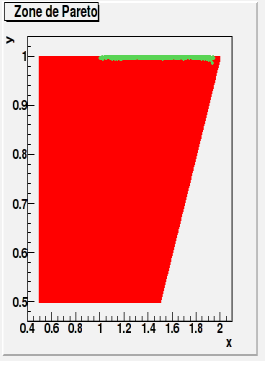
- Functions prototype : **void myFunction (Double_t *x, Double_t *y)**
- Constraints prototype : **void myConstraint (Double_t *x, Int_t &g)**

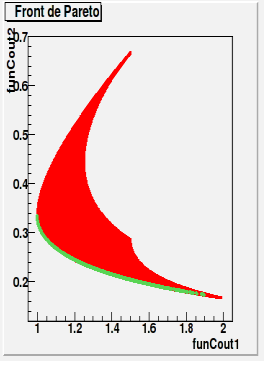
Problem of the bar:

$$f_1(x,y) = (x-1)^2 + (y-1)^2 + 1$$
$$f_2(x,y) = (x^2 + y^2 + 1)^{-1}$$
$$g(x,y) = x - y - 1$$



```
TDataServer * tds = new TDataServer();
tds->addAttribute( new TAttribute("x", 0.5, 2.0));
tds->addAttribute( new TAttribute("y", 0.5, 1.0));
VizirMulti *vzrmulti = new VizirMulti(tds, 1000);
vzrmulti->addCost(funCout1);
vzrmulti->addCost(funCout2);
vzrmulti->addHardConstraint(contHard1);
vzrmulti->optimize();
```





"Sensitivity" library

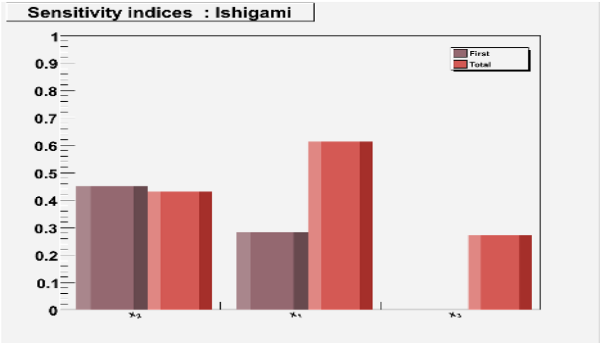
Perform a sensitivity analysis between the X and Y attributes

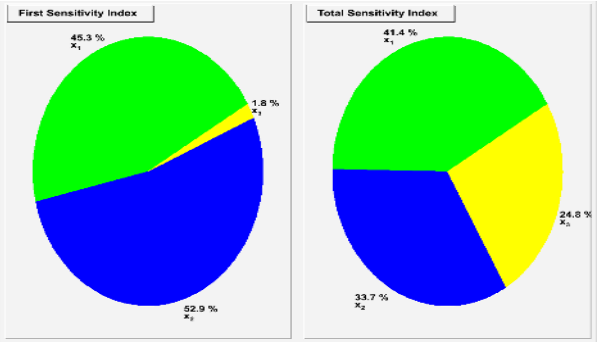
- Regression methods
  - Pearson (values)
  - Spearman (Rank)
- "Screening" method like **Morris** method
- "Sobol" indexes
  - *"Brute-force"* Method (First Order)
  - FAST : "Fourier Amplitude Sensitivity Test" (First Order)
  - RBD : "Random Balance Design" (First Order)
  - "Sobol" Methods
    - ⋆ First Order with "Sobol93/Jansen99/Saltelli02"
    - ⋆ Total Order with "Homma96/Jansen99/Sobol07"

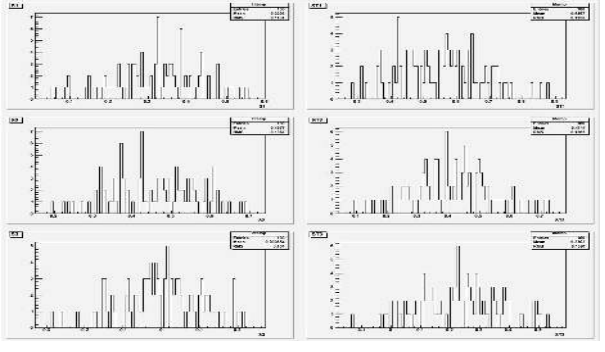
Application : "Ishigami" function

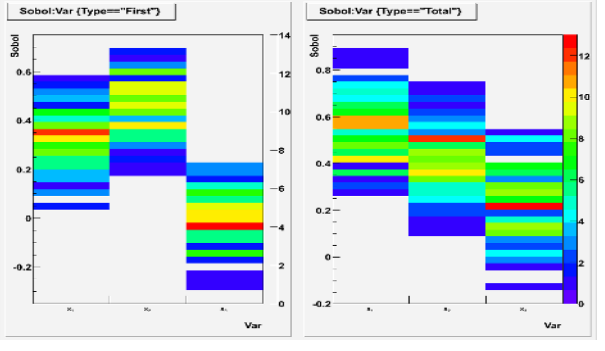
ISHIGAMI benchmark : Analytical function of $R^3 \to R$ ($x_{i=1,2,3} \sim Unif[-\pi, \pi]$)

$$y = \sin x_1 + A \sin^2 x_2 + B x_3^4 \sin x_1 \qquad A = 7, B = 0.1$$





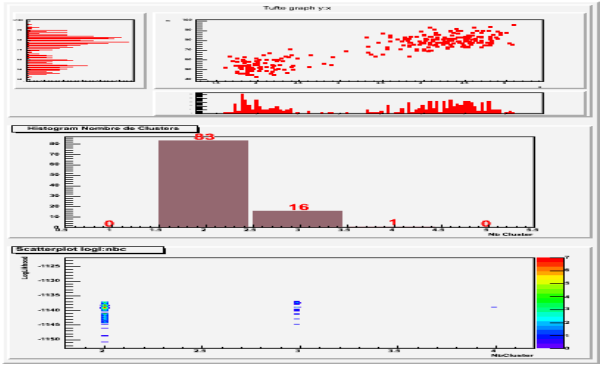


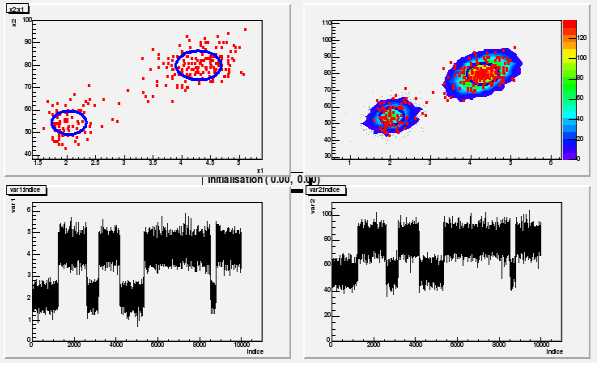


"UncertModeler" library

Identify a law (Probability Density Function) from a database

- Parametric law
  - QQ-plot
  - Goodness-of-fit methods: Shapiro-Wilks, Kolmogorov-Smirnov, Cramer-von Mises, Anderson-Darling
- Gaussian mixture **MixMod** (GPL)





Environment Variables

- Shell **CSH**

> *source /export/opt/URANIE/v2.3.0/uranie.cshrc*

Options of the **ROOT** command

- -b : Batch mode (without graphics windows)
- -l : Without the Splash window
- -q : Exit ROOT after execute the macro
- -n : Without loading the *rootlogon.C / rootlogoff.C* files

CINT : the C++ interpreter mode

- .q : Exit
- .h : Help
- .x : Execute the macro file
- .L : Load the macro file
- .class : API of the class
- .! : Shell command

Exemples

Command line

```
$> root myMacro.C
$> root -l myMacro.C
$> root -b -q myMacro.C > myMacro.log
```

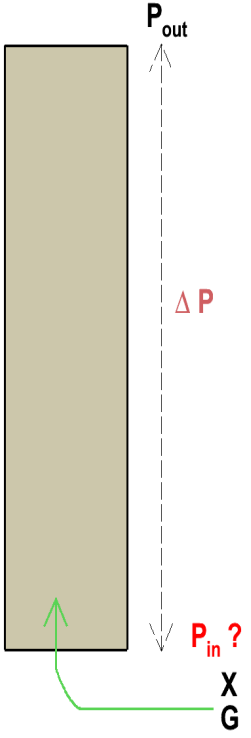
CINT mode

```
$> root

root[] > .x myMacro.C
root[] > .L myMacro.C
root[] > myMacro();
```

Documentation

- Manuals
  - User Manual
    http://root.cern.ch/root/doc/RootDoc.html
  - Reference Guide
    http://root.cern.ch/root/htmldoc/ClassIndex.html
- Tutorials
  http://root.cern.ch/root/html/tutorials
- Howto
  http://root.cern.ch/drupal/content/howtos
- Trainings
  - ROOT training for beginners
    http://caeinfo.in2p3.fr/root/Formation/Formation.html
  - ROOT Educational Resources at Fermilab
    http://www-root.fnal.gov/root/

Flica IV application



- **Measurements** in the data set :
  - Outlet Pressure P_{out}
  - Mass flow rate G
  - Quality X
- **Models** for Pressure losses

$$\Delta P = \Delta P_f + \Delta P_g + \Delta P_a$$

$$\Delta P_f := f_{2\varphi} * F_{iso} * \frac{G^2}{2\rho D_h}$$

$$F_{iso} := a_t Re^{b_t}$$

- **Model Parameters** :
  - $(a_t, b_t) \in [0.20, 0.40] \times [0.15, 0.35]$

- "**geometrie**" : the geometry of the facility
- "**cdlim_70**" : boundary conditions data set $(P_{out}, G, X, \Delta P_{exp})$
- "**post-traitement**" : specify the output variables to store
- "**entete**" : values of the model parameters
  - ⋆ $(a_t, b_t) \in [0.20, 0.40] \times [0.15, 0.35]$

```
MODELE.'FROTTEMENT'.'ISO_ATURB' = 0.284 ;
MODELE.'FROTTEMENT'.'ISO_BTURB' = 0.245 ;
```

  - ⋆ Compute several cases (max 61) :

```
CALCUL.'SUITE' = INITABLE: 1 11 21;
```

  - ⋆ Use the *TEST_SEQUENCE* macro :

```
#define TEST_SEQUENCE "post-traitement"
```

Flica IV application : Input files 2/2

```
*  ------------------------------------------------------------------
*  |                                                                |
*  |                        Post-processing FLICA-4                 |
*  |________________________________________________________________|

BLOC TEST:SEQUENCE ;
...

*  ------------------------------------------------------------------
*  |                                                                |
*  |                            Post-traitment                      |
*  |________________________________________________________________|

INDEX = CALCUL.'NUMERO';

* Parametres At and Bt for the isothermal frottement model
AT           = MODELE.'FROTTEMENT'.'ISO_ATURB' ;
BT           = MODELE.'FROTTEMENT'.'ISO_BTURB' ;
...
P_SORTIE     = TABLE_PL.1.NZ1 + TABLE_PV.1.NZ1 ;
P_ENTREE     = TABLE_PL.1.1   + TABLE_PV.1.1   ;
DELTAP_TOT   = ABS: ( P_SORTIE - P_ENTREE ) ;

P_EXP   = LIMITE.'VALEURS_P'.INDEX   ;
G_EXP   = LIMITE.'VALEURS_G'.INDEX   ;
X_EXP   = LIMITE.'VALEURS_X'.INDEX   ;
DELTAP_EXP   = LIMITE.'EXPERIENCE_DP'.INDEX *  1.E5    ;

SORTIE_2D: 1 93 'COMPLETE'
   'X' ( INITABLE: ( REEL: INDEX ) )
   'Y' ( INITABLE: AT )
       ( INITABLE: BT )
       ( INITABLE: P_ENTREE )
       ( INITABLE: P_SORTIE )
       ( INITABLE: DELTAP_TOT )
       ( INITABLE: P_EXP )
       ( INITABLE: G_EXP )
       ( INITABLE: X_EXP )
       ( INITABLE: DELTAP_EXP )
       1 ;

FIN TEST:SEQUENCE ;
```

Flica IV application : Output file 1/1

– Flica IV command line :

```
f4 -vPUB -sESSAI $PWD 10
```

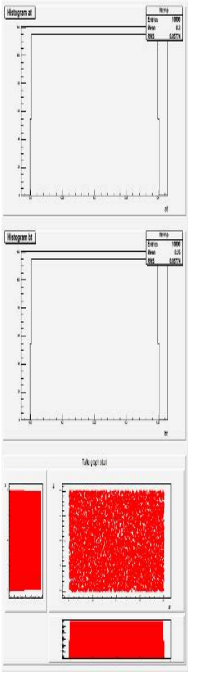
– All the output variables are stored in the **"fort.93"** file :

```
# NOMBRE COLONNES = 10 NOMBRE LIGNES = 001
# FONCTION = COMPLETER
  1.000000E 00  1.840000E 01  2.000000E 01  7.048290E 06  7.034600E 06  1.369040E 04  7.034600E 01  4.983000E 01  4.213000E 02  1.327500E 04
# FIN DES DONNEES
# NOMBRE COLONNES = 10 NOMBRE LIGNES = 001
# FONCTION = COMPLETER
  1.100000E 01  1.840000E 01  2.000000E 01  7.028272E 06  6.998600E 06  2.967183E 04  6.998600E 01  1.502200E 02  1.385900E 01  2.312900E 04
# FIN DES DONNEES
# NOMBRE COLONNES = 10 NOMBRE LIGNES = 001
# FONCTION = COMPLETER
  2.100000E 01  1.840000E 01  2.000000E 01  7.029664E 06  6.994600E 06  3.506458E 04  6.994600E 01  1.249400E 02  2.713400E 01  2.441100E 04
# FIN DES DONNEES
```

Uranie macro : macro "sampler.C"

- Objective:
  - ⋆ Specify the problem (only input attributes)
  - ⋆ Generate a *LHS* sampling with size $nS = 10$

```
{
  // Specify the problem
  TString sJDD = "entete";
  TDataServer *tds = new TDataServer("tdsf4", "tds for F4 code");
  TUniformDistribution *attat = new TUniformDistribution("at", 0.20, 0.40);
  attat->setFileKey(sJDD, "MODELE.'FROTTEMENT'.'ISO_ATURB'");
  tds->addAttribute( attat);
  TUniformDistribution *attbt = new TUniformDistribution("bt", 0.15, 0.35);
  attbt->setFileKey(sJDD, "MODELE.'FROTTEMENT'.'ISO_BTURB'");
  tds->addAttribute( attbt);

  // Generate the Design of Experiments (DoE)
  Int_t nS = 10000;
  TSampling *samp = new TSampling(tds, "lhs", nS);
  samp->generateSample();

  // startViewer
  tds->startViewer();

  // Graphs
  TCanvas * c = new TCanvas("c1", "c1",15,36,564,898);
  c->Divide(1,3);
  c->cd(1); tds->draw("at");
  c->cd(2); tds->draw("bt");
  c->cd(3); tds->drawTufte("bt:at");
  c->SaveAs("appliUranieFlicaCiseSampler.png");
}
```



– Objective : Evaluate the FLICA IV code on the LHS design.

```
{
  ...
  samp->generateSample();

  // Specify the computation code
  TOutputFileRow *fout = new TOutputFileRow("ESSAI/fort.93");
  fout->addAttribute(new TAttribute("on"));
  fout->addAttribute(new TAttribute("oat"));
  fout->addAttribute(new TAttribute("obt"));
  fout->addAttribute(new TAttribute("ope"));
  fout->addAttribute(new TAttribute("ops"));
  fout->addAttribute(new TAttribute("odp"));
  fout->addAttribute(new TAttribute("ip"));
  fout->addAttribute(new TAttribute("ig"));
  fout->addAttribute(new TAttribute("ix"));
  fout->addAttribute(new TAttribute("idpexp"));

  TCode *codef4 = new TCode(tds, "f4 -vPUB -sESSAI %D 10 >> /dev/null");
  codef4->setReferenceDirectory(gSystem->pwd());
  codef4->addInputFile("cdlim_70");
  codef4->addInputFile("geometrie");
  codef4->addInputFile("post-traitement");
  codef4->addOutputFile(fout);

  // Evaluate the DoE on the computation code
  TLauncher *lan = new TLauncher(tds, codef4);
  lan->setSave();
  lan->setClean();
  lan->run();

  //Save the data in ASCII file
  tds->exportData("_f4_launching.dat");
}
```

> *more "_f4_launching.dat"*

```
2.2983468e-01 1.5060738e-01 1.0e+00 2.298347e-01 ... 7.0346e+01 4.9830e+01 4.2130e-02 1.3275e+04 1.00e+00
2.2983468e-01 1.5060738e-01 1.1e+01 2.298347e-01 ... 6.9986e+01 1.5022e+02 1.3859e-01 2.3129e+04 1.00e+00
2.2983468e-01 1.5060738e-01 2.1e+01 2.298347e-01 ... 6.9946e+01 1.2494e+02 2.7134e-01 2.4411e+04 1.00e+00

1.9625798e-01 2.3149332e-01 1.0e+00 1.962580e-01 ... 7.0346e+01 4.9830e+01 4.2130e-02 1.3275e+04 2.00e+00
1.9625798e-01 2.3149332e-01 1.1e+01 1.962580e-01 ... 6.9986e+01 1.5022e+02 1.3859e-01 2.3129e+04 2.00e+00
1.9625798e-01 2.3149332e-01 2.1e+01 1.962580e-01 ... 6.9946e+01 1.2494e+02 2.7134e-01 2.4411e+04 2.00e+00

1.8783251e-01 2.0148748e-01 1.0e+00 1.878325e-01 ... 7.0346e+01 4.9830e+01 4.2130e-02 1.3275e+04 3.00e+00
1.8783251e-01 2.0148748e-01 1.1e+01 1.878325e-01 ... 6.9986e+01 1.5022e+02 1.3859e-01 2.3129e+04 3.00e+00
1.8783251e-01 2.0148748e-01 2.1e+01 1.878325e-01 ... 6.9946e+01 1.2494e+02 2.7134e-01 2.4411e+04 3.00e+00
```

Column:

- 1 and 2 : (a_t, b_t) of the sample
- 3 : the index variable
- 4 and 5 : (a_t, b_t) viewed by the FLICA IV code
- and so on for the output attributes
- last one : the iterator attribute (internal)

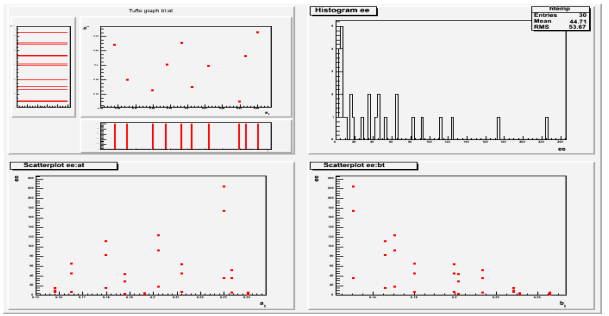
Uranie macro : file "rootlogon.C"

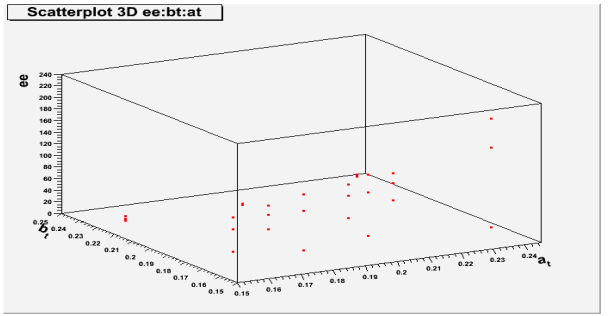
```
{
  using namespace URANIE::DataServer;
  using namespace URANIE::Sampler;
  using namespace URANIE::Launcher;
  using namespace URANIE::Modeler;
  using namespace URANIE::Optimizer;
  using namespace URANIE::Sensitivity;

  gROOT->LoadMacro("fonctions.C");

  gStyle->SetPalette(1);
  gStyle->SetLabelSize(0.02,"xyz");
  gStyle->SetMarkerColor(2);
  gStyle->SetMarkerStyle(7);
  gStyle->SetMarkerSize(1.1);

  TDataServer * tds= new TDataServer();
  //tds->fileDataRead("_f4_launching.dat");
  tds->fileDataRead("../data/flica4.dat");
  tds->addAttribute("ee", "TMath::Abs((idpexp-odp))*100.0/idpexp");

  tds->getAttribute("at")->SetTitle("a_{t}");
  tds->getAttribute("oat")->SetTitle("a_{t}");
  tds->getAttribute("bt")->SetTitle("b_{t}");
  tds->getAttribute("obt")->SetTitle("b_{t}");
  tds->getAttribute("odp")->SetTitle("#Delta P_{Tot}");
  tds->getAttribute("idpexp")->SetTitle("#Delta P^{Exp}");
  tds->getAttribute("ix")->SetTitle("Quality");
  tds->getAttribute("ix")->setUnity("%");

}
```

Compute the *error* attribute $ee = \frac{|\Delta P^i_{exp} - \Delta P^i_{mod}(a_t, b_t)|}{\Delta P^i_{exp}} * 100.0$

– Uranie macro *"Analyse1.C"* :
> *root -l analyse1.C*

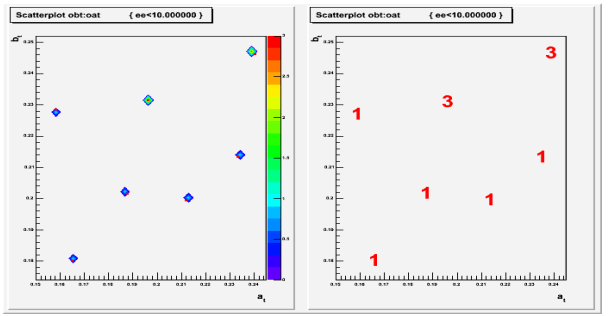
```
{
  // Graph
  TCanvas * c= new TCanvas();
  c->Divide(2,2);

  c->cd(1); tds->drawTufte("bt:at");
  c->cd(2); tds->draw("ee");
  c->cd(3); tds->draw("ee:at");
  c->cd(4); tds->draw("ee:bt");
  c->SaveAs("appliUranieFlicaCiseAnalyse1c1Database.png");

  c= new TCanvas();
  tds->draw("ee:bt:at");
  c->SaveAs("appliUranieFlicaCiseAnalyse1c2Database.png");
}
```

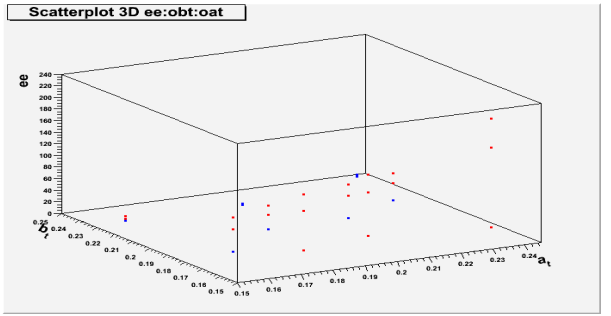




– Uranie macro *"Analyse2.C"* :
> *root -l analyse2.C*

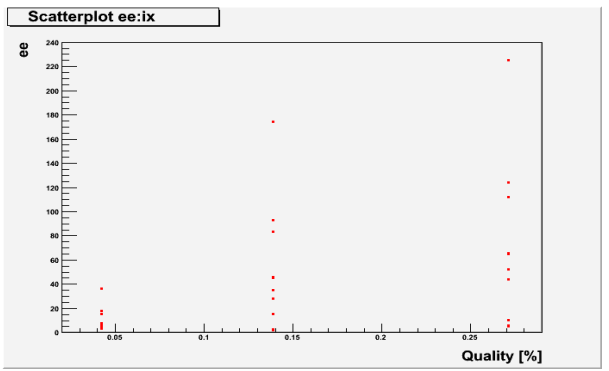
```
{
  Double_t dThresold = 10.0;

  // Visualisation
  TCanvas *c = new TCanvas("c2_1", "Analysis 2 - My first canvas");
  c->Clear(); c->Divide(2);

  c->cd(1); tds->draw("obt:oat",Form("ee<%f",dThresold), "contz");tds->draw("obt:oat",Form("ee<%f",dThresold),"same");
  c->cd(2); tds->draw("obt:oat", Form("ee<%f",dThresold), "text");
  c->SaveAs("appliUranieFlicaCiseAnalyse2c1Database.png");

  c = new TCanvas("c2_2", "Analysis 2 - My second canvas");
  gr3DSelectPct(tds, "ee:obt:oat", Form("ee<%4.2f", dThresold));
  c->SaveAs("appliUranieFlicaCiseAnalyse2c2Database.png");
}
```

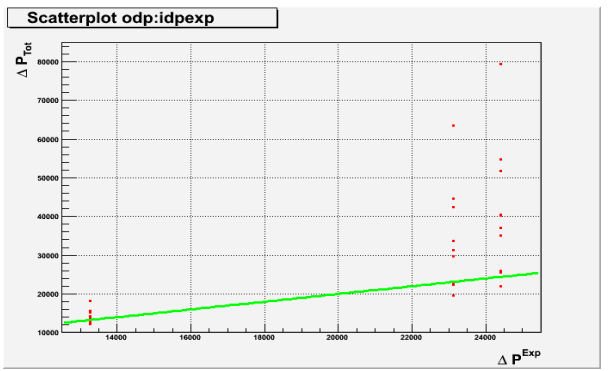




– Uranie macro *"Analyse3.C"* :
> *root -l analyse3.C*

```
{
  TCanvas *c = new TCanvas("c3_1", "Analysis 3 - My second canvas");
  tds->Draw("ee:ix","","p");
  c->SaveAs("appliUranieFlicaCiseAnalyse3c1Database.png");

  c = new TCanvas("c3_2", "Analysis 3 - My second canvas");
  tds->Draw("odp:idpexp");
  TF1 * bis= new TF1("bb", "x",8000.0, 50000.0);
  c->SetGrid(1,1);
  bis->SetLineColor(3);
  bis->Draw("same");
  c->SaveAs("appliUranieFlicaCiseAnalyse3c2Database.png");
}
```





Big DataBase

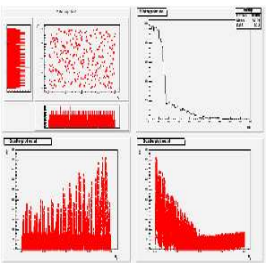
- Big database (DoE with 300 patterns and 61 cases)
- File *"rootlogon.C"*

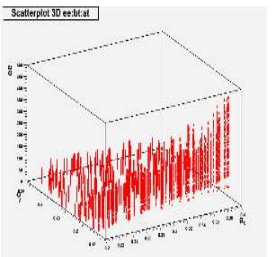
```
TDataServer * tds= new TDataServer();
//tds->fileDataRead("_f4_launching.dat");
tds->fileDataRead("../data/flica4.dat");
```

Uranie macro : macro "analyse1.C"

> *root -l analyse1.C*

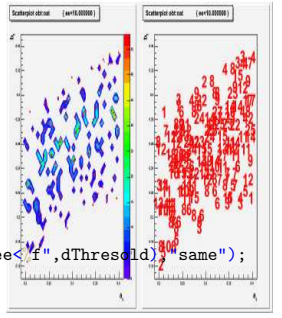
```
{
  // Graph
  TCanvas * c= new TCanvas();
  c->Divide(2,2);

  c->cd(1); tds->drawTufte("bt:at");
  c->cd(2); tds->draw("ee");
  c->cd(3); tds->draw("ee:at");
  c->cd(4); tds->draw("ee:bt");
  c->SaveAs("appliUranieFlicaCiseAnalyse1c1Database.png");

  c= new TCanvas();
  tds->draw("ee:bt:at");
  c->SaveAs("appliUranieFlicaCiseAnalyse1c2Database.png");
}
```

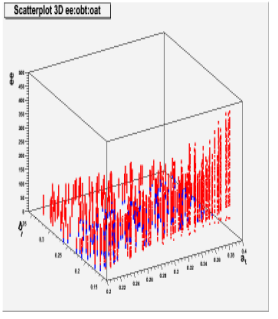




> *root -l analyse2.C*
– file "analyse2.C"

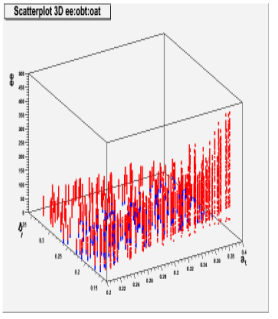
```
{
  Double_t dThresold = 10.0;

  // Visualisation
  TCanvas *c = new TCanvas("c2_1", "Analysis 2 -
My first canvas");
  c->Clear(); c->Divide(2);

  c->cd(1); tds->draw("obt:oat",Form("ee<%f",dThresold), "contz");tds->draw("obt:oat",Form("ee<%f",dThresold),"same");
  c->cd(2); tds->draw("obt:oat", Form("ee<%f",dThresold),
"text");
  c->SaveAs("appliUranieFlicaCiseAnalyse2c1Database.png");

  c = new TCanvas("c2_2", "Analysis 2 - My second canvas");
  gr3DSelectPct(tds, "ee:obt:oat", Form("ee<%4.2f",
dThresold));
  c->SaveAs("appliUranieFlicaCiseAnalyse2c2Database.png");
}
```





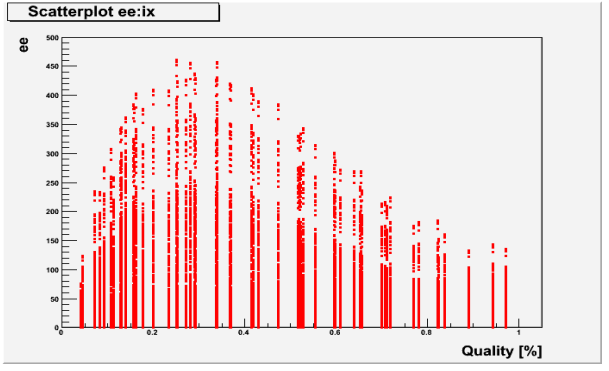
– file "fonctions.C"

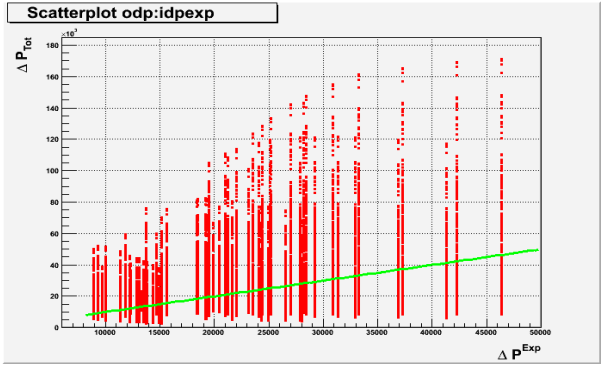
```
//! Graph 1D, 2D or 3D with a different color for patterns which verify select
/*!
  \param ntd (TNtupleD *) The TNtupleD which
contains the data
  \param sVars (TString) the list of attributes
to make the graph
  \param sSelect (TString) the select string

*/
void gr3DSelectPct(TDataServer *tds, TString
sVars, TString sSelect)
{
  tds->getTuple()->SetMarkerSize(1.5);
  tds->getTuple()->SetMarkerColor(2);
  tds->Draw(sVars);
  tds->getTuple()->SetMarkerSize(1.5);
  tds->getTuple()->SetMarkerColor(4);
  tds->Draw(sVars,sSelect,"same");
  tds->getTuple()->SetMarkerColor(2);

}
```



Uranie macro : macro "analyse3.C"

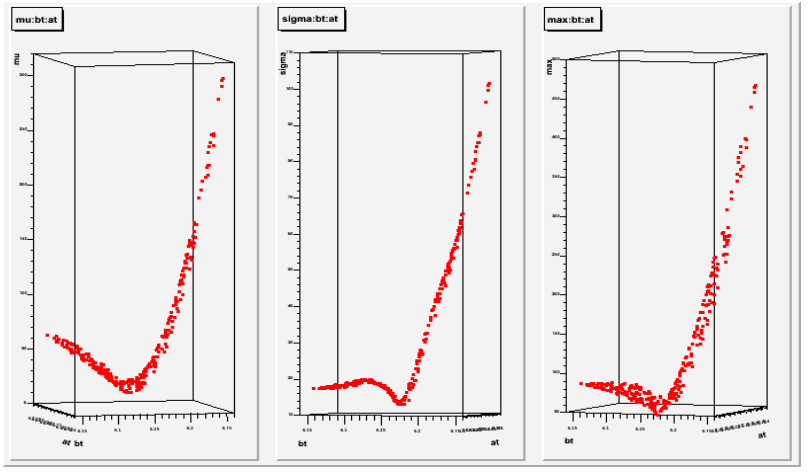
```
{
  TCanvas *c = new TCanvas("c3_1", "Analysis 3 - My second canvas");
  tds->Draw("ee:ix","","p");
  c->SaveAs("appliUranieFlicaCiseAnalyse3c1Database.png");

  c = new TCanvas("c3_2", "Analysis 3 - My second canvas");
  tds->Draw("odp:idpexp");
  TF1 * bis= new TF1("bb", "x",8000.0, 50000.0);
  c->SetGrid(1,1);
  bis->SetLineColor(3);
  bis->Draw("same");
  c->SaveAs("appliUranieFlicaCiseAnalyse3c2Database.png");
}
```





– Create the 3 output attributes $\mu = \mu_{ee}$, $\sigma = \sigma_{ee}$ and $max = \max ee$ with the $n = 61$ cases where

$$\mu_{ee}(a_t, b_t) = \frac{1}{n} \sum_{i=1}^{n} \frac{|\Delta P^i_{exp} - \Delta P^i_{mod}(a_t, b_t)|}{\Delta P^i_{exp}} * 100.0$$



Uranie macro : macro "buildTarget.C" *(cont.)*

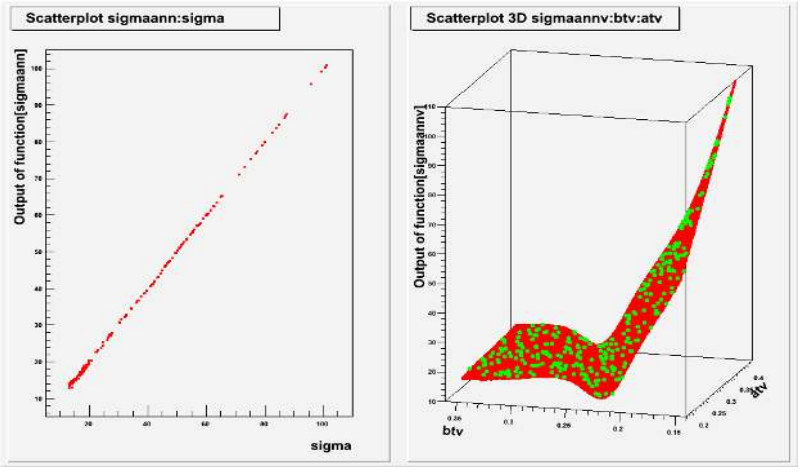
```
{
  tds->computeStatistic(tds->getIteratorName());
  Int_t nS = tds->getAttribute(tds->getIteratorName())->getMaximum();
  cout << " ** nS[" << nS << "]" << endl;

  TNtupleD * ntd = new TNtupleD("ntdmean", "the mean", "at:bt:mu:sigma:max");
  for (Int_t iS=0; iS<nS; iS++) {
    tds->setSelect(Form("%s==%d", tds->getIteratorName(), iS+1));
    tds->computeStatistic("at:bt:ee");
    ntd->Fill( tds->getAttribute("at")->getMean(),
        tds->getAttribute("bt")->getMean(),
        tds->getAttribute("ee")->getMean(),
        tds->getAttribute("ee")->getStd(),
        tds->getAttribute("ee")->getMaximum()
        );
  }
  TCanvas *c = new TCanvas();
  c->Divide(3);
  c->cd(1);  ntd->Draw("mu:bt:at");
  c->cd(2);  ntd->Draw("sigma:bt:at");
  c->cd(3);  ntd->Draw("max:bt:at");
  c->SaveAs("appliUranieFlicaCiseBuildTargetmusigmamax.png");

  TDataServer * tdsntd = new TDataServer("tdsntd", "the true");
  tdsntd->loadTree(ntd);
  tdsntd->exportData("../data/flica4_ann.dat");
}
```

Uranie macro : macro "buildANN.C"

– Create a surrogate model for μ or σ or max with the 300 patterns

– Validate the ANN with a DoE of 60000 patterns

> *root -l buildANN.C*



Uranie macro : macro "buildANN.C" *(cont.)*

```
{
  TDataServer * tds= new TDataServer("tdsann", "tds to build ANN");
  tds->fileDataRead("../data/flica4_ann.dat");
  tds->draw("sigma:bt:at");

  Int_t nHH = 6;

  // Builds a surrogate model (Artificial Neural Networks) from the DataBase
  TANNModeler* tann = new TANNModeler(tds, Form("at:bt,%d,sigma", nHH));
  tann->setNormalization(TANNModeler::kMinusOneOne);
  tann->setLog();
  tann->setFcnTol(1e-5);
  tann->train(5, 5, "test");
  tann->exportFunction("c++", "uranie_ann_sigma_flica4.C","annsigmaFLica4");
  tann->exportFunction("pmml", "uranie_ann_flica4.pmml","annsigmaFLica4");

  gROOT->LoadMacro("uranie_ann_sigma_flica4.C+");

  // evaluate the surrogate model on the database
  TLauncherFunction * tlf = new TLauncherFunction(tds, annsigmaFLica4, "at:bt", "sigmaann");
  tlf->run();

  TCanvas *c = new TCanvas("c1", "c1",15,36,1028,700);
  c->Divide(2);
  c->cd(1); tds->Draw("sigmaann:sigma");

  TDataServer *tdsv = new TDataServer("tdsf4val", "tds for validation F4 ANN");
  tdsv->addAttribute( new TUniformDistribution("atv", 0.20, 0.40));
  tdsv->addAttribute( new TUniformDistribution("btv", 0.15, 0.35));

  Int_t nVal = 60000;
  TSampling *samp = new TSampling(tdsv, "lhs", nVal);
  samp->generateSample();
  TLauncherFunction * tlfv = new TLauncherFunction(tdsv, annsigmaFLica4, "atv:btv", "sigmaannv");
  tlfv->run();
  c->cd(2); tdsv->draw("sigmaannv:btv:atv");
  tds->getTuple()->SetMarkerColor(kGreen);
  tds->getTuple()->SetMarkerStyle(8);
  tds->getTuple()->SetMarkerSize(1.25);
  tds->draw("sigma:bt:at","","same");
  tds->getTuple()->SetMarkerColor(kRed);
  c->SaveAs("appliUranieFlicaCiseBuildANNsigma.png");
}
```

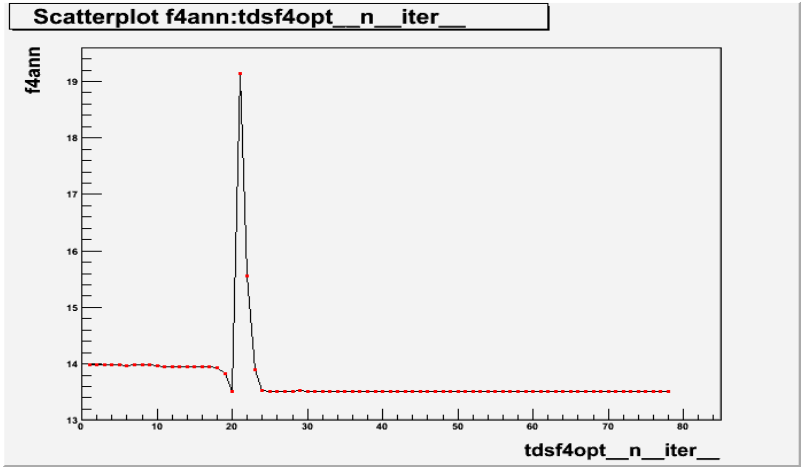
```
#define ActivationFunction(sum)         ( tanh(sum) )
void annsigmaFLica4(double *param, double *res)
{
  ////////////////////////////
  //
  //     ******************************************
  //     ** Uranie  Tue Jul  5 13:05:37 2011
  //     **   Version : v2.3/1
  //     **      Date : Thu Feb 10, 2011
  //     ******************************************
  //
  //
  //     ******************************************
  //     ** tdsann **
  //     **
  //     ** tds to build ANN **
  //     ******************************************
  //
  //    INPUT : 2
  //             at ( at )
  //             bt ( bt )
  //
  //    HIDDEN : 6
  //
  //    OUTPUT : 1
  //             sigma ( sigma )
  //
  ////////////////////////////
  int nInput   = 2;
  int nOuput   = 1;
  int nHidden  = 6;
  const int nNeurones  = 9;
  double annsigmaFLica4_act[nNeurones];

  // --- Pretraitment of the inputs and outputs
  double annsigmaFLica4_minInput[] = {
  0.200579, 0.150431,
  };
  double annsigmaFLica4_minOuput[] = {
  12.9786,   };
  double annsigmaFLica4_maxInput[] = {
  0.39939, 0.349826,
  };
  double annsigmaFLica4_maxOuput[] = {
  101.209,   };

  // --- Values of the weights
  double annsigmaFLica4_valW[] = {
  3.4659, -0.313576, 0.793925, -1.10482, 4.05277,
  4.71573, -1.86074, 0.170525, -1.04831, 0.256658,
```

```
<?xml version="1.0"?>
<PMML version="3.0" xmlns="http://www.dmg.org/PMML-3_0">
  <Header copyright="texte copyright" description="texte description">
    <Application name="Uranie" version="2.3/1"/>
    <Annotation>date Thu Feb 10, 2011</Annotation>
  </Header>
  <DataDictionary>
    <DataField name="at" displayName="at" optype="continuous" dataType="float"/>
    <DataField name="bt" displayName="bt" optype="continuous" dataType="float"/>
    <DataField name="sigma" displayName="sigma" optype="continuous" dataType="float"/>
  </DataDictionary>
  <NeuralNetwork functionName="regression" numberOfLayers="2" modelName="annsigmaFLica4_0">
    <MiningSchema>
      <MiningField name="at" usageType="active"/>
      <MiningField name="bt" usageType="active"/>
      <MiningField name="sigma" usageType="predicted"/>
    </MiningSchema>
    <NeuralInputs numberOfInputs="2">
      <NeuralInput id="0">
        <DerivedField optype="continuous" dataType="float">
          <NormContinuous field="at">
            <LinearNorm orig="2.005792e-01" norm="-1"/>
            <LinearNorm orig="3.993902e-01" norm="1"/>
          </NormContinuous>
        </DerivedField>
      </NeuralInput>
      <NeuralInput id="1">
        <DerivedField optype="continuous" dataType="float">
          <NormContinuous field="bt">
            <LinearNorm orig="1.504310e-01" norm="-1"/>
            <LinearNorm orig="3.498257e-01" norm="1"/>
          </NormContinuous>
        </DerivedField>
      </NeuralInput>
    </NeuralInputs>
    <NeuralLayer activationFunction="tanh" numberOfNeurons="6">
      <Neuron id="2" bias="-3.684059e-01">
        <Con from="0" weight="-1.237614e+00"/>
        <Con from="1" weight="4.756162e+00"/>
      </Neuron>
      <Neuron id="3" bias="-1.854366e+00">
        <Con from="0" weight="-9.607521e-02"/>
        <Con from="1" weight="-1.409266e+00"/>
      </Neuron>
      <Neuron id="4" bias="-1.932427e+00">
        <Con from="0" weight="3.297699e+00"/>
        <Con from="1" weight="-2.437880e-01"/>
      </Neuron>
      <Neuron id="5" bias="-1.879300e+00">
        <Con from="0" weight="8.541586e-02"/>
```

Uranie macro : macro "optimizeWithANN.C"

– Make an optimization with the surrogate model for μ or σ or max

> *root -l optimizeWithANN.C*



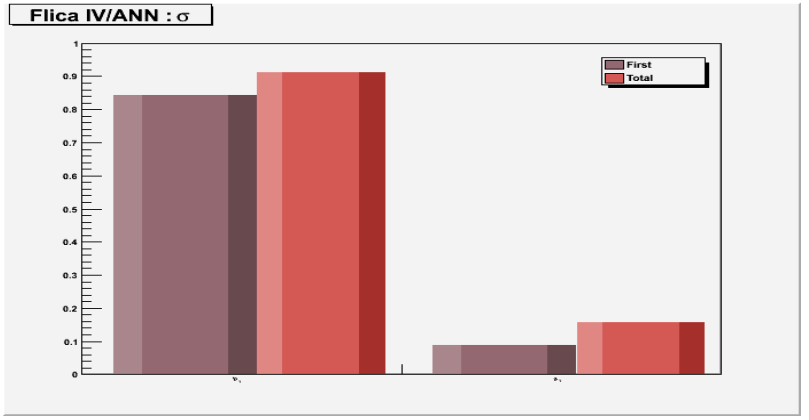
Uranie macro : macro "optimizeWithANN.C" *(cont.)*

```
{

  gROOT->LoadMacro("uranie_ann_sigma_flica4.C+");

  TDataServer *tds = new TDataServer("tdsf4opt", "tds for Optimize F4 code");
  tds->addAttribute( new TAttribute("at", 0.20, 0.40));
  tds->addAttribute( new TAttribute("bt", 0.15, 0.35));

  tds->getAttribute("at")->setDefaultValue(0.30);
  tds->getAttribute("at")->setStepValue(0.01);
  tds->getAttribute("bt")->setDefaultValue(0.25);
  tds->getAttribute("bt")->setStepValue(0.01);

  // Create an TOptimizer object from TDataServer and an anlystical function
  TOptimizer * topt = new TOptimizer(tds, annsigmaFLica4,"","f4ann");
//   topt->setMethod(TOptimizer::kSimplex);
  topt->setTolerance(1e-5);
  topt->setPrintLevel(5);
  topt->setMaxIterations(10);
  topt->setMaxFunctionCalls(100);
  topt->optimize("same");
  tds->exportData("_tds_rosenbrock_.dat");
  gPad->SaveAs("appliUranieFlicaCiseOptimizewithANN.png");

}
```

Uranie macro : macro "sobolWithANN.C"

– Compute the *Sobol* indexes by the **Sobol** method with a DoE of 60000 patterns

> *root -l sobolWithANN.C*



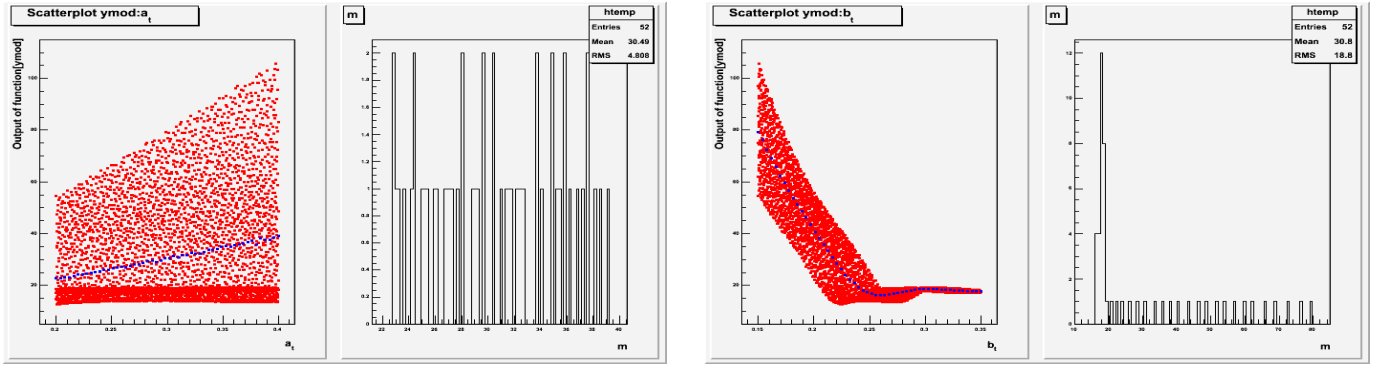
Uranie macro : macro "sobolWithANN.C" *(cont.)*

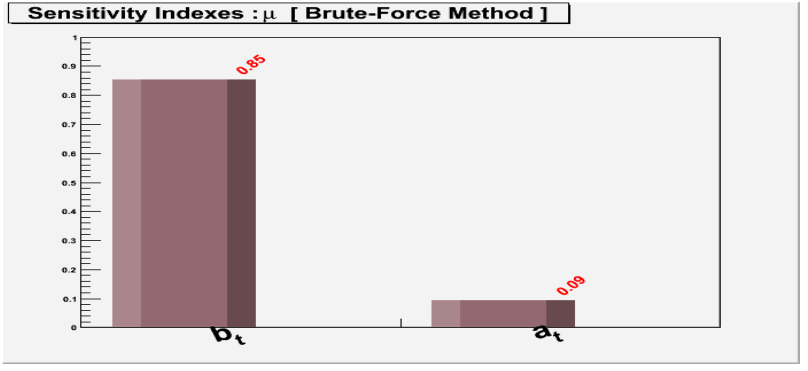
```
{
  gROOT->LoadMacro("uranie_ann_sigma_flica4.C+");

  TDataServer *tds = new TDataServer("tdsf4opt", "tds for Optimize F4 code");
  tds->addAttribute( new TUniformDistribution("a_{t}", 0.20, 0.40));
  tds->addAttribute( new TUniformDistribution("b_{t}", 0.15, 0.35));

  Int_t ns = 60000;
  TSobol * tsobol = new TSobol(tds, annsigmaFLica4, ns);
  tsobol->computeIndexes();

  TCanvas *cc = new TCanvas("canhist", "histgramme");
  tsobol->drawIndexes("Flica IV/ANN : #sigma", "", "nonewcanv,hist,all");
  cc->SaveAs("appliUranieFlicaCiseSobolwithANNsigma.png");
}
```

Uranie macro : macro "sobolWithANNBF.C"

– Compute the First *Sobol* indexes by the **Brute-Force** method

> *root -l sobolWithANNBF.C*





Uranie macro : macro "sobolWithANNBF.C" *(cont.)*

```
void sobolWithANNBF(Int_t nCond = 50, Int_t nbins = 50)
{
  cout << endl << " ******************************************************" << endl;
  cout << " ** sensitivityBrutForceMethod ANN nbins[" << nbins << "] nCond[" << nCond << "]" << endl;
  cout << " **" << endl;

  gROOT->LoadMacro("uranie_ann_sigma_flica4.C+");

  TDataServer *tds = new TDataServer("tdsf4opt", "tds for Optimize F4 code");
  tds->addAttribute( new TUniformDistribution("a_{t}", 0.20, 0.40));
  tds->addAttribute( new TUniformDistribution("b_{t}", 0.15, 0.35));

  Int_t nvar = tds->getNAttributes();
  cout << " ** nX[" << nvar << "]" << endl;

  Int_t nS = nbins*nvar*nCond;
  cout << " ** nS[" << nS << "]" << endl;

  TQMC * sam = new TQMC(tds, "halton", nS);
  sam->generateSample();

  // Create a TLauncherFunction from a TDataServer and an analytical function
  // Rename the outpout attribute "ymod"
  TLauncherFunction * tlf = new TLauncherFunction(tds, annsigmaFLica4,"","ymod");
  // Evaluate the function on all the design of experiments
  tlf->run();

  TCanvas *c = new TCanvas();
  tds->computeStatistic("ymod");
  tds->draw("ymod");
  Double_t dstdy = tds->getAttribute("ymod")->getStd();
  Double_t svary = dstdy * dstdy;
  cout << " ** ymod : std[" << dstdy << "] vary[" << svary  << "]" << endl;

  gStyle->SetOptStat(1);

  // Tempory TTree for Histogram visualisation
  Double_t valSobolCrt;
  Char_t   sName[12];
  TTree *tt = new TTree("sobolforcebrut","sensibilité sobol MonteCarlo Force brute");
  tt->Branch("Var",(void *)sName,"Var/C");
  tt->Branch("Sobol",&valSobolCrt,"Sobol/D");
  tt->SetMarkerColor(kRed);
  tt->SetMarkerStyle(7);
  tt->SetMarkerSize(1.75);

  TCanvas *c = new TCanvas();
  c->Divide(2);
  c->cd(1);
```

Conclusions

- Introduced the ROOT framework
- Presented librairies of the Uranie platform
- Applications of Uranie :
  - ⋆ Generating a Design Of Experiments (DoE)
  - ⋆ Launching an external code
  - ⋆ Building a surrogate model (*Artificial Neurals Networks* - **ANN**)
  - ⋆ Optimization of model parameters with ANN
  - ⋆ Sensitivity Analysis with ANN