Présentation Uranie v1.1

Séminaire IMPEC

F. Gaudier

CEA/DEN/DANS/DM2S/SFME/LGLS

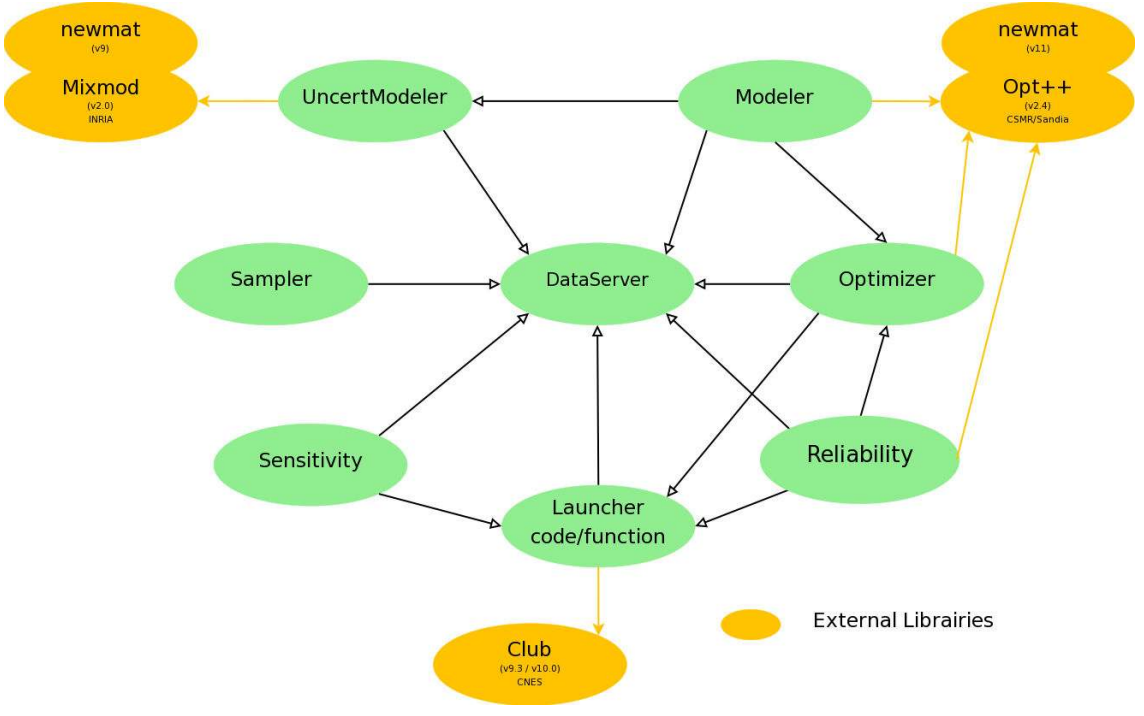
CEA - Cadarache, le 13/10/2008

URANIE : CEA/DEN Uncertainty Platform
URANIE : Fonctional diagram
URANIE : Fonctional diagram
URANIE : Graphical User Interface
URANIE - XML User Interface
URANIE : Batch mode
Projects using URANIE

URANIE : CEA/DEN Uncertainty Platform

- ROOT for data analysis, CLUB (CNES), MIXMOD (INRIA), OPT++ (Sandia)
- QT for GUI
- 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)
  - Surrogate models (Polynomial, Artificial Neural Networks, Splines)
  - Sensitivity analysis (Pearson, Spearmann, Sobol, Fast, Morris)
  - Optimization (MetaModeling, Genetic Algorithms)
  - Computing distribution

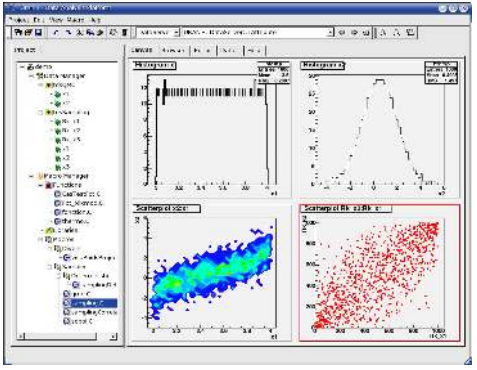
URANIE : Fonctional diagram



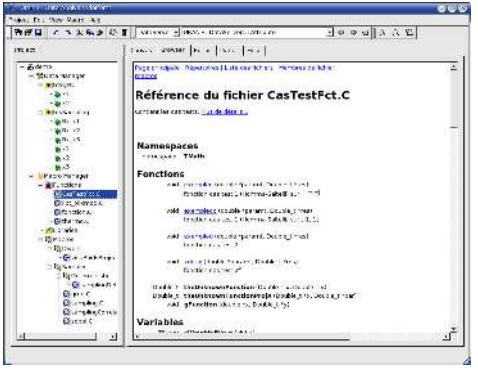
URANIE : Fonctional diagram

Libraries	Lines (*.h, *.cxx)	Classes
DataServer	13 000	21
Sampler	10 000	14
Launcher	5 000	10
Modeler	9 000	9
Optimizer	4 000	6
Sensitivity	3 000	6
UncertModeler	2 000	5
Sous-Total	46 000	70
IHM	13 000	34
editor	1 000	7
cppeditor	300	2
Sous-Total	14 300	43
Total	60 300	113

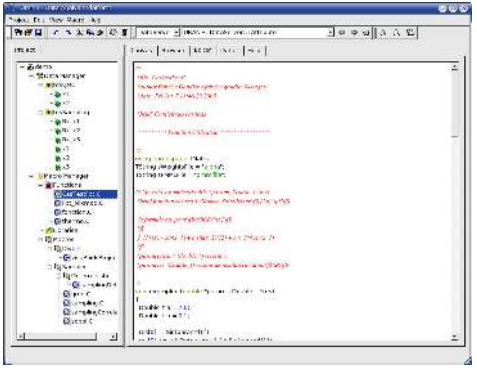
URANIE : Graphical User Interface



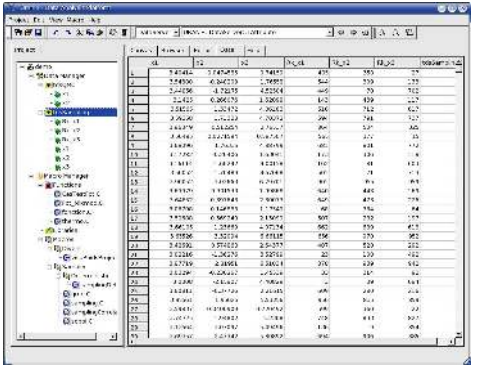
Visualization



User Help



Editor



Spreadsheet

URANIE - XML User Interface

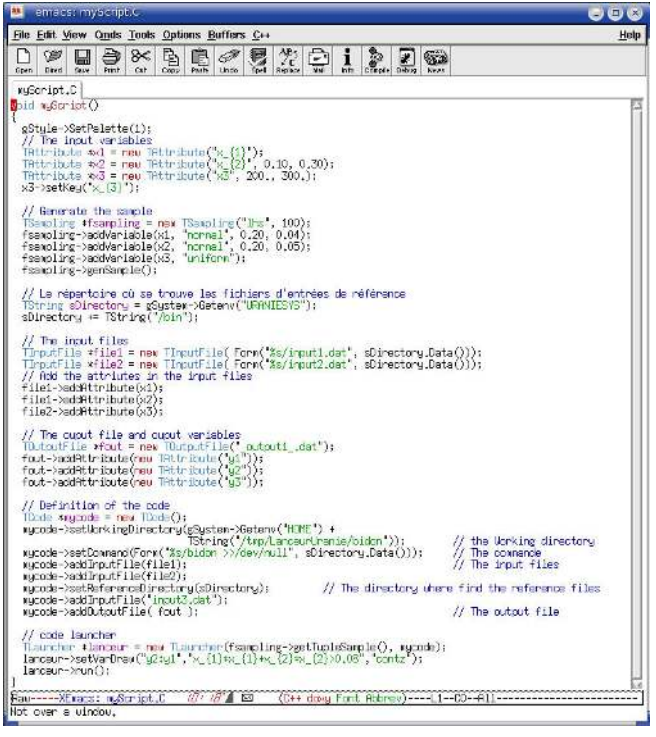
XML file (problem_uranie.xml)

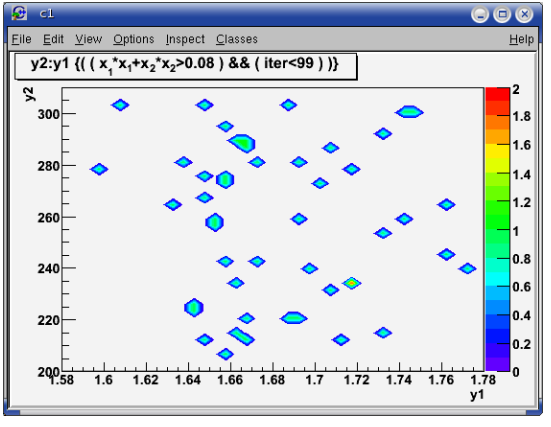
```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE Problem SYSTEM "/home/uranie/tools/share/uranie/uranie.dtd" >
<Problem>
    <Header name="Etude" title="projet GENTR">
        <Application name="uranie" version="0.4"/>
    </Header>
    <DataDictionary>
        <DataField name="x1" law="uniform" min="0.5" max="1.5"/>
        <DataField name="x2" law="normal" mean="2.5" std="0.25"/>
    </DataDictionary>
    <Sampler method="SRS" N="1500" export="data/sampler_SRS_1500.dat"/>
    <Sampler method="LHS" N="1000" export="data/sampler_LHS_1000.dat"/>
</Problem>
```

```
uranie -s problem_uranie.xml
```

URANIE : Batch mode

> root myScript.C





URANIE

"DataServer"

"Sampler" . . .

"Launcher" . .

"Modeler" . . .

"Optimizer" .

"UncertModel

"Sensitivity" .

Plan de . . .

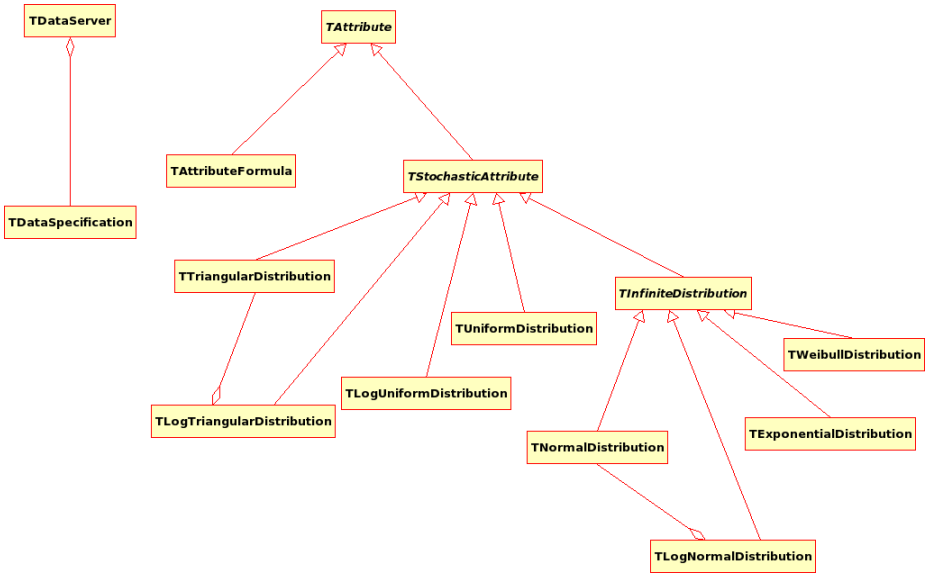
Projects using URANIE

- LEONAR tool for severe accidents in french nuclear reactor (**CEA-EDF**)
- Dosimetry computation in french nuclear reactor (**CEA-EDF**)
- Opus project : Meteor code (**CEA**)
- CIVA tool : "Non Destructive Testing" with (**CEA/DRT**)
- Sensitivity Analysis for Cathare code (**CEA/Areva TA**)
- 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

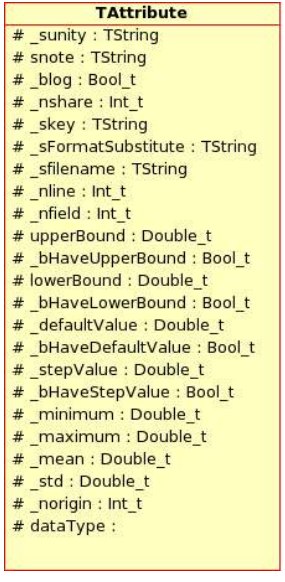
"DataServer" library - Features

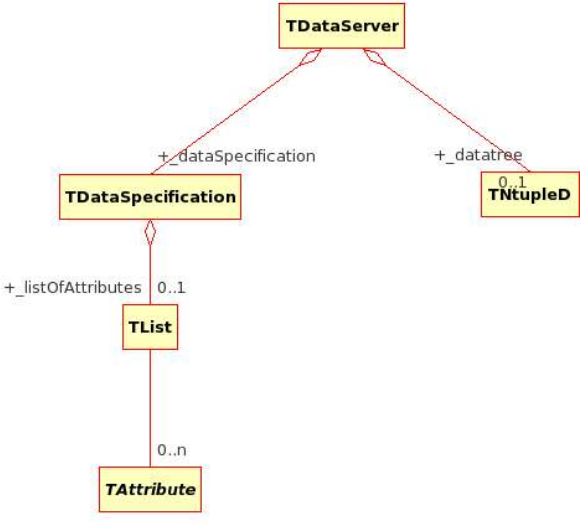
1. Management of the attributes (∼ variables)
   - create/transform attributes
   - Load data from external files/formats (ASCII,TTREE,SQL)
2. Graphs and treatments specific to uncertainties
3. Specification of problems XML



"DataServer" library - Attributes

- An **Attribute** contains :
  Name, Title, Unity, Min/Max/Default/Step values, Key/File
- A random variable is an attribute + a law defined by parameters
- All the attributes are stored in a **TDataServer**





Management of the attributes : Load data

- Load data from external files (ASCII, TTREE, SQL, ...)

```
using namespace URANIE::DataServer;
{
TDataServer *tds = new TDataServer();
tds->fileDataRead("geyser.dat");
tds->addAttribute("cd", "sqrt(x2) * x1");
tds->draw("cd:x1");
}
```

```
...
TDataServer *tds = new TDataServer();
tds->ntupleDataRead("hsimple.root","ntuple","px*py:*:py*px", "px*px+py*py<2.0");
tds->draw("py:px");
...
```

"DataServer" module - URANIE ASCII file format

#TITLE: geyser data

#NAME: geyser

#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

...

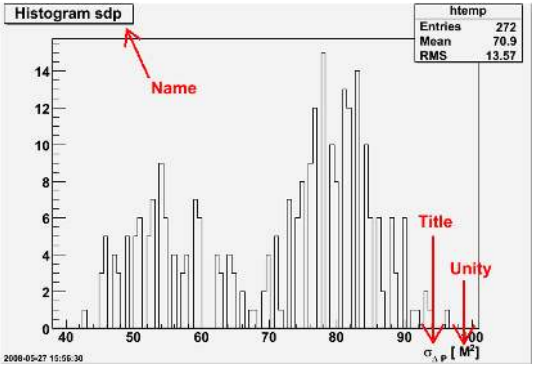


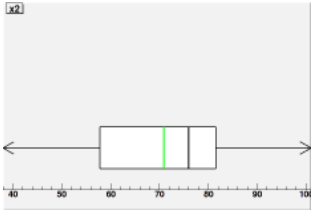
Figure 2.1 tds->draw("sdp");

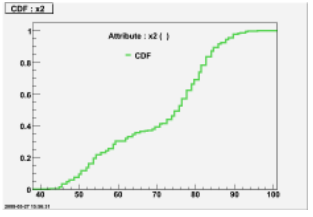
Only the `"#COLUMN_NAMES:"` line is obligatory

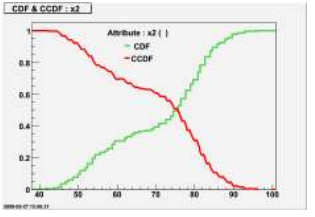
WARNING : the empty line between the header and the matrix data

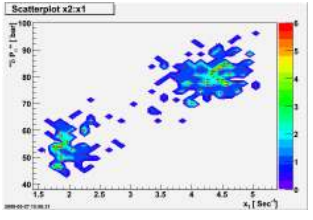
”DataServer” library - statistical graphs

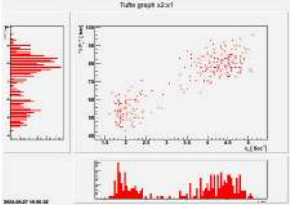
```
tds->drawBoxPlot(”x2”);
tds->drawCDF(”x2”,”x1<3.0”);
tds->drawCDF(”x2”,”x1<3.0”,”ccdf”);
tds->drawScatterplot(”x2:x1”);
tds->drawTufte(”x2:x1”);
tds->drawProfile(”x2:x1”,””,”same”);
```

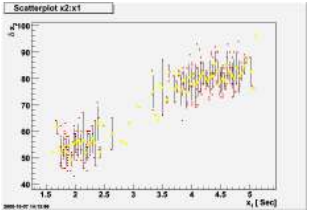












Histogramm - Number of bins

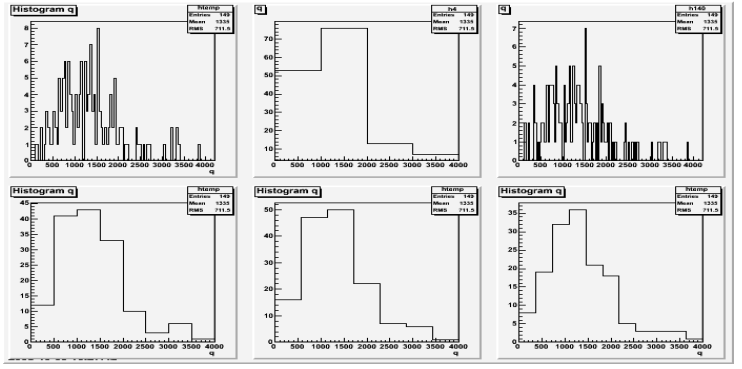
– Defined in the **".rootrc"** file

```
# Default histogram binnings for TTree::Draw().
Hist.Binning.1D.x: 100
```

– Exist in **R** :

1. **Sturges** $N_{bin} = log_2(n) + 1$
2. **Scott** $N_{bin} = (x_{max} - x_{min}) * \sqrt[3]{n}/3.5\hat{\sigma}_x$
3. **Freedman & Diaconis** $N_{bin} = (x_{max} - x_{min}) * \sqrt[3]{n}/2 * (Q_x^{0.75} - Q_x^{0.25})$

```
tds->draw("x", "", "nclass=root");
tds->draw("x", "", "nclass=sturges");
tds->draw("x", "", "nclass=fd");
tds->draw("x", "", "nclass=scott");
```



Treatment - Correlation

Correlations matrix (**Pearson, Spearmann**) : $\rho_{XY} = \frac{I\!E[\ (X-\mu_X)(Y-\mu_Y)\ ]}{\sigma_X \sigma_Y}$

```
TDataServer * tds = new TDataServer("tds", "Sampling");
tds->addAttribute(new TUniformDistribution("x1", 3., 4.));
tds->addAttribute(new TNormalDistribution("x2", 0.5, 1.5));

TSampling *sampling = new TSampling(tds, "lhs", 20);
sampling->setUserCorrelation("x1", "x2", 0.789);
sampling->generateSample();

tds->computeCorrelationMatrix("x1:x2");
tds->computeCorrelationMatrix("x1:x2","","rank");
```

Variables	x1	x2
x1	1	
x2	0.775	1

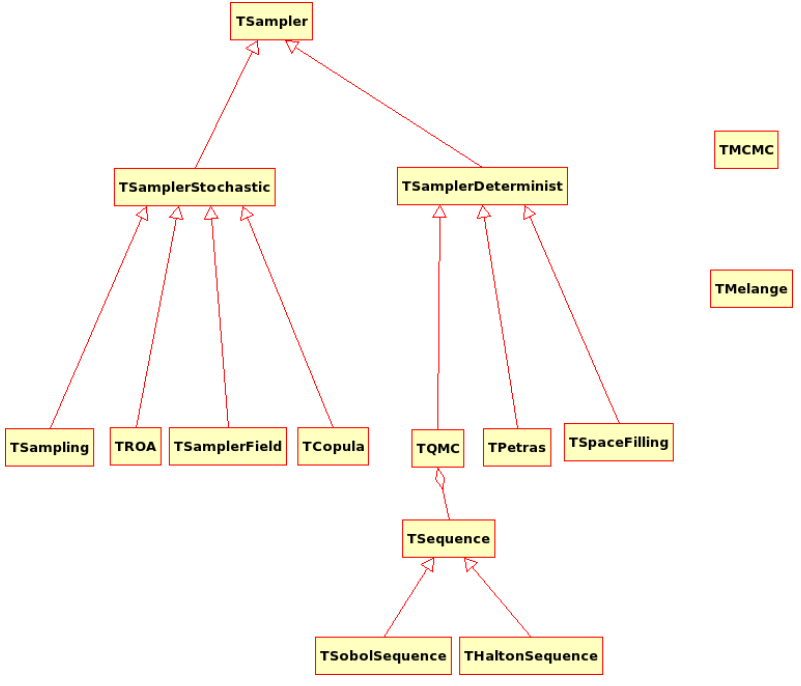
Pearson

Rank	x1	x2
x1	1	
x2	0.782	1

Spearmann

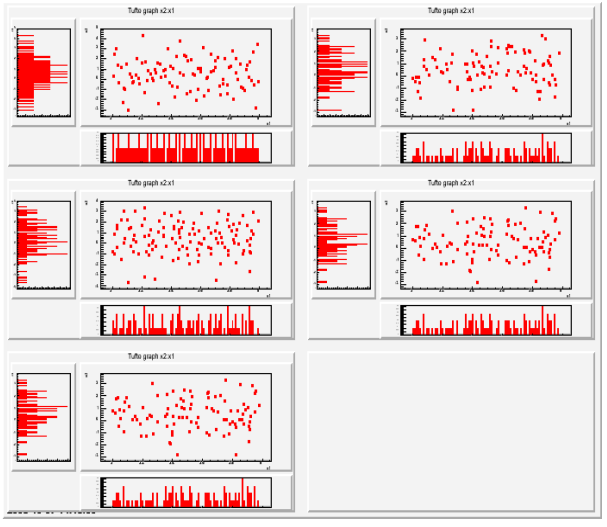
"Sampler" library

Stochastic TSampling
Deterministic Sampling
Generate a sampling from a **TDataServer** object

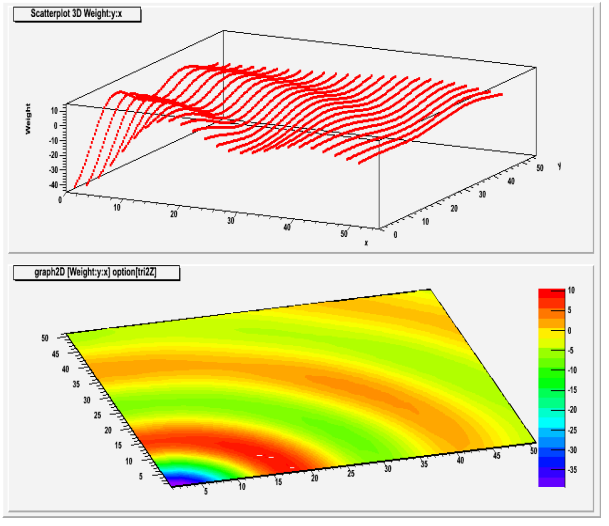


Stochastic TSampling

- "Simple Random Sampling" SRS / "Latin Hypercube Sampling" **LHS**
  - ⋆ Rank correlations
- "Random Orthogonal Array" ROA
- Archimedian Copulas (Gumbel, Clayton, Frank)
- Random Field
- "Markov Chain Monte Carlo" (MCMC) for Gaussian mixture



x_1 and x_2 uniform - Size 100



Gaussian Field

Deterministic Sampling

- quasi Monte-Carlo Sequences (Halton, Sobol)
- Petras
- Space Filling Design

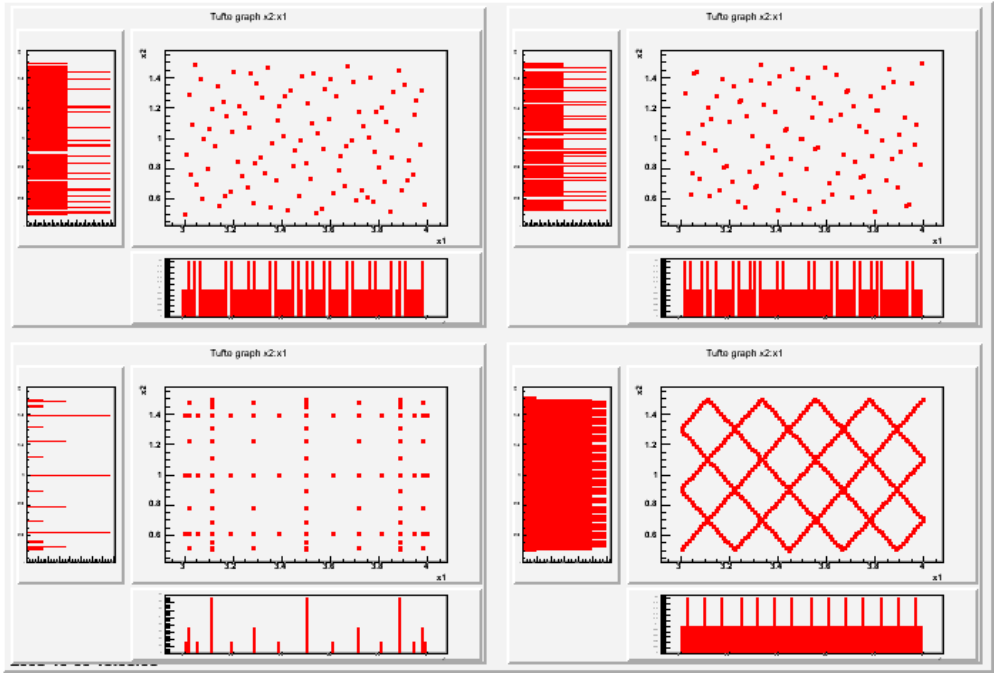
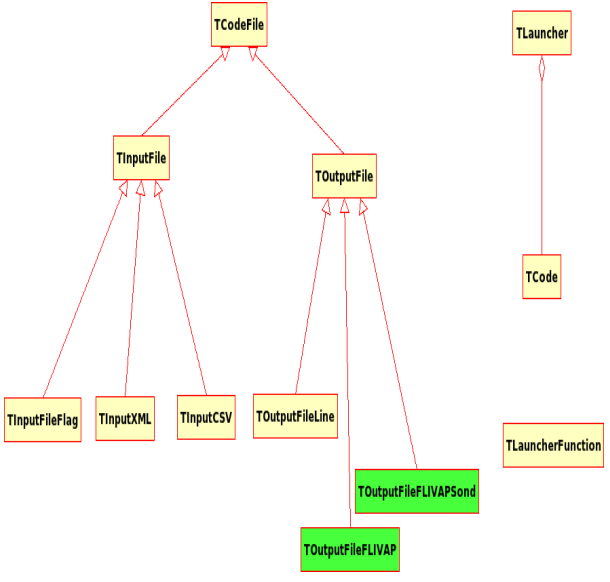


Figure 3.1 x_1 and x_2 uniform - Size 100

Input File : **" Key - Value"** format
Input File with "flag"
Distribution CCRT
Feature : Distribute the model evaluations (sequential, cluster)



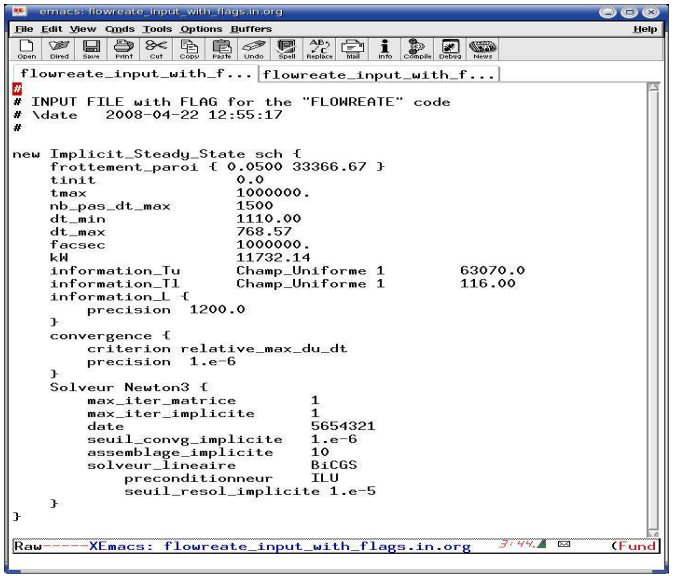
Input File : ” **Key - Value**” format

```
#
#
# fichier d'entrée 1 du code \b bidon
# \date   Tue Sep  9 09:56:41 2003
# les 3 premiers paramètres
#

npar = 10 ;

x_{0} = 1.2345 ;
 x_{1} = 0.206846449673 ;
x_{2} = 0.197665744126 ;
```

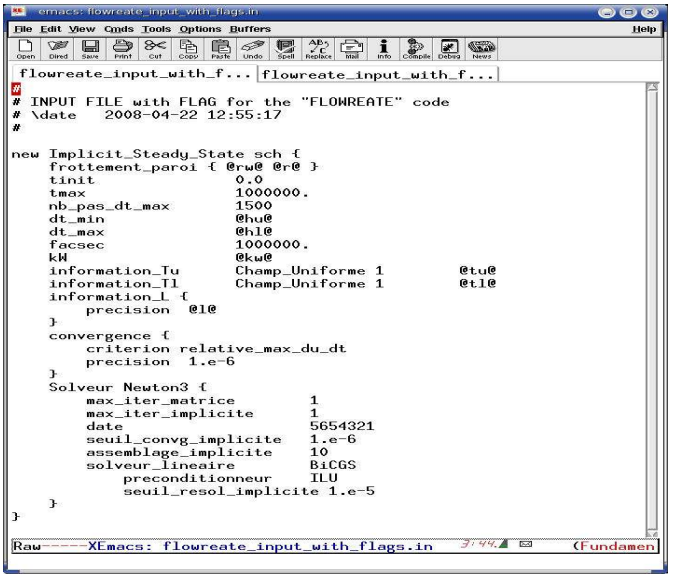
```
TAttribute *x1 = new TAttribute(”x_{1}”, 0.20, 0.04);
TAttribute *x2 = new TAttribute(”x2”, 200., 300.);
x2->setKey(”x_{2}”);

TInputFile *file1 = new TInputFile(”input1.dat”);

file1->addAttribute(x1);
file1->addAttribute(x2);
```

Hypothesis : unicity of the key

Input File with "flag"



Original file



User Flag file

```
attrw->setKey("myfile.in", "@rw@");
```

Hypothesis : Not unicity of the key but intervention of the user

Distribution CCRT

```
#BSUB -n 10
#BSUB -J FlowreateSampling
#BSUB -o FlowreateSampling.out
source /home/cont002/gaudier/uranie-platine.cshrc
rm -rf FlowreateSampling.out
root -l -q lanceurFLOWREATE_SAMPLING.C
```

> bsub < BsubFile

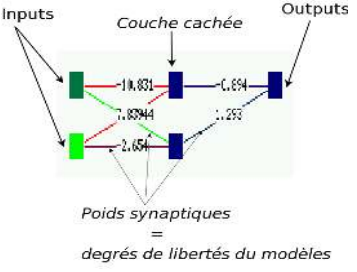
```
<?xml version="1.0" encoding="iso-8859-1"?>
<main>
 <machine-list>
   <machine env-file="/home/cont002/gaudier/uranie-platine.cshrc"
      work-directory="/work/cont002/gaudier/testKERNELSALOME_is205980">platine</machine>
   <machine env-file="/home/gaudier/uranie.cshrc"
      work-directory="/work/gaudier/testKERNELSALOME">awa</machine>
 </machine-list>
 <ref-directory>/home/gaudier/tmp/testuranie/testKERNELSALOME</ref-directory>
 <nb-processes>64</nb-processes>
 <input-file>lanceurFLOWREATE_SAMPLING.C</input-file>
 <input-file>flowreate_input_with_keys.in</input-file>
 <input-file>flowrateborhole.dat</input-file>
 <output-file>_flowreate_sampler_launcher_.dat</output-file>
 <command>rm -f platine.error.log</command>
 <command>rm -f platine.output.log</command>
 <command>root -b -l -q lanceurFLOWREATE_SAMPLING.C</command>
</main>
```

> uranieDistrib flowreate.xml platine

"Modeler" library - features

"Modeler" library - resampling method : *Bootstrap*
Application : *Sinus Cardinal*
Create an analytical function between Y and X

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



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

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

- Export function in C, C++, Fortran
  using for code calibration, propagation of uncertainties, ⋯
- Save in **PMML** format : "*Predictive Model Markup Language*" (DMG)

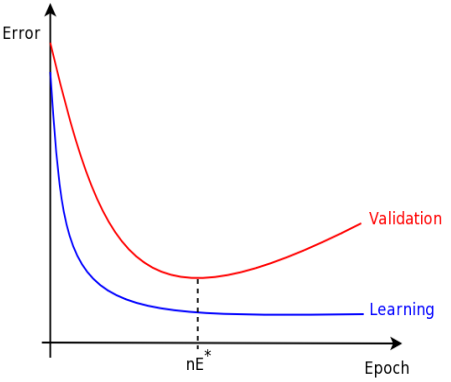
"Modeler" library - resampling method : *Bootstrap*

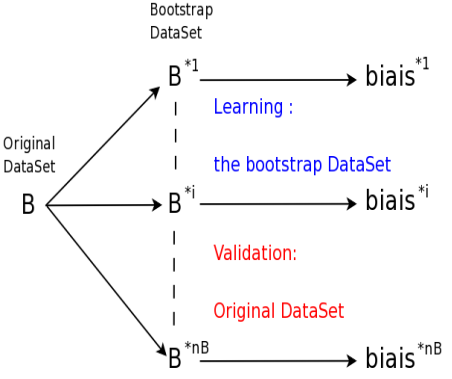
- **Context**: Small data base
- **Goal** : Estimate the number of epoch
  1. Use all the data in the learning process
  2. Find the optimal number of epoch $nE^\star$
  3. Estimate the validation error ϵ_V
- **Tool** : Resampling method (Bootstrap)

$$biais \quad \delta = \epsilon_V - \epsilon_L$$

$$\bar{\delta}^{\star,\alpha} = \frac{1}{nB^{\star,\alpha}} \sum \delta^{\star,i} \quad \forall \delta^{\star,i} \in [q_\alpha, q_{1-\alpha}]$$

$$\hat{\epsilon}_V = \epsilon_L + \bar{\delta}^{\star,\alpha}$$

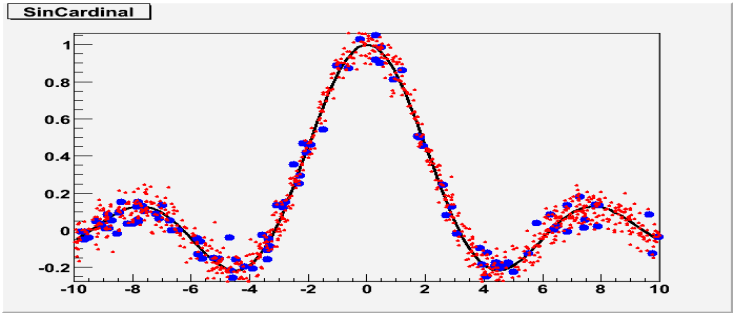


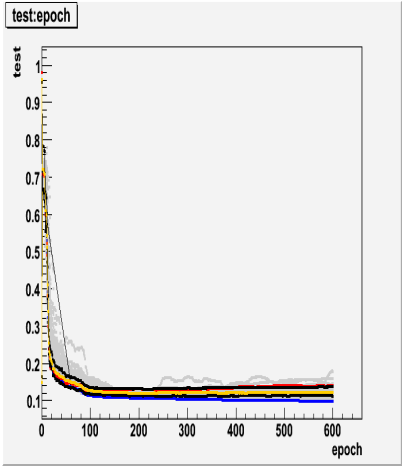


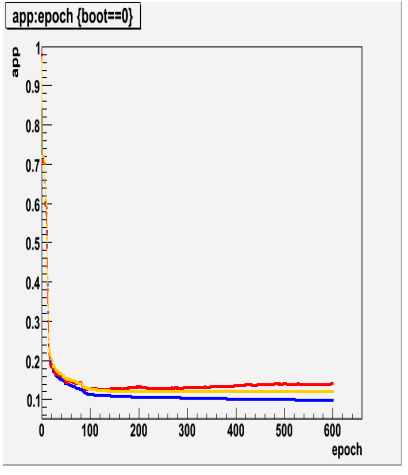
Application : *Sinus Cardinal*

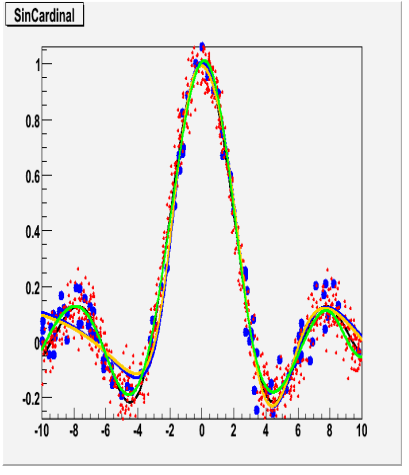
$$f(x) = \frac{\sin|x|}{|x|} + \epsilon$$

- Noise $\epsilon \sim \mathcal{N}(0., 0.06)$
- Learning : 100 (blue)
- Validation : 900 (red)
- Bootstrap : $nB = 50$









"Optimizer" library . . .

. . .

. . .

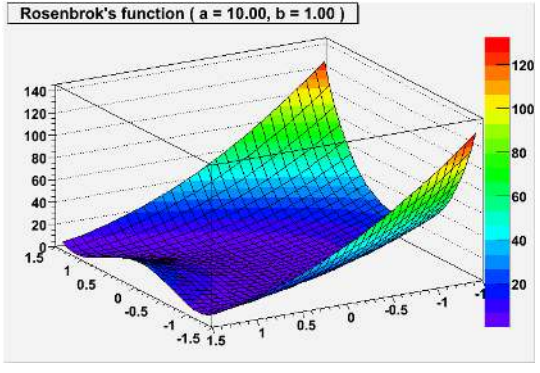
Identification des paramètres de modèles

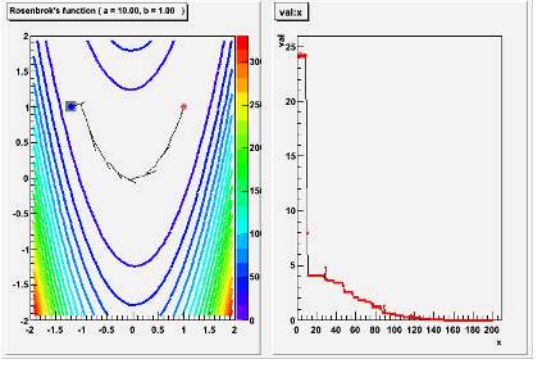
"Optimizer" library (1/3)

- librairie **Minuit2** de **ROOT**
  - ⋆ Prototype des fonctions : **void myFunction (Double_t *param, Double_t *res)**
  - ⋆ un objet URANIE::Launcher::TCode

Rosenbrock : $f(x,y) = a(y-x^2)^2 + b(1-x)^2$ avec $a = 100.$ et $b = 1.$

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





Optimisation non-linéaire sous contraintes avec **opt++**

- Posséde plusieurs algorithmes d'optimisation (Direct, gradient, Newton, ...)
- Mode DLL pour les prototypes **opt++**

```
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)
```

Exemple:

```
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");
```

- Mode interprété pour le prototype URANIE (en cours de développement)

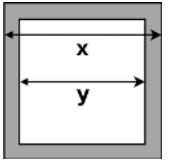
"Optimizer" library (3/3)

Optimisation multicritères par Algorithme Génétique : **Vizir**

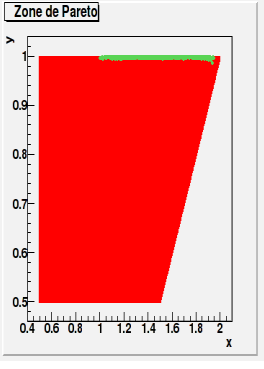
- Prototype des fonctions : **void myFunction (Double_t *param, Double_t *res)**
- Prototype des contraintes : **void myConstraint (Double_t *param, Int_t &res)**

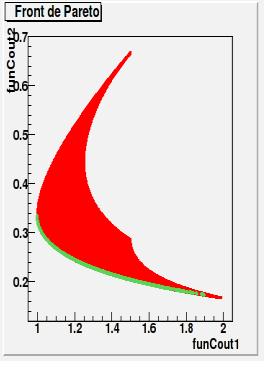
Problème de la barre:

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



```
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();
```

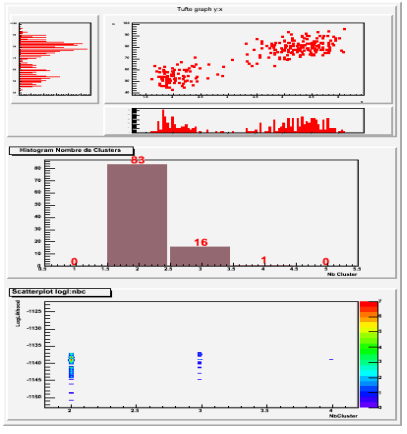


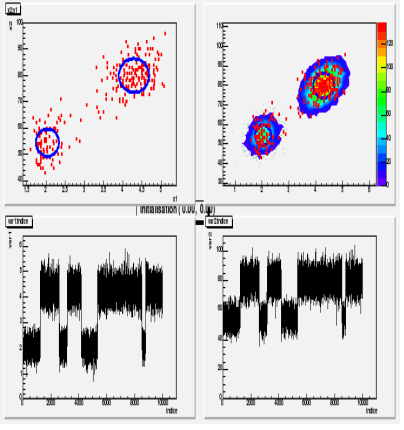


"UncertModeler" library

Identify a law (Probability Density Function) from a data base.

- Parametric law
  - ⋆ QQ-plot
- Gaussian mixture **MixMod** (GPL)

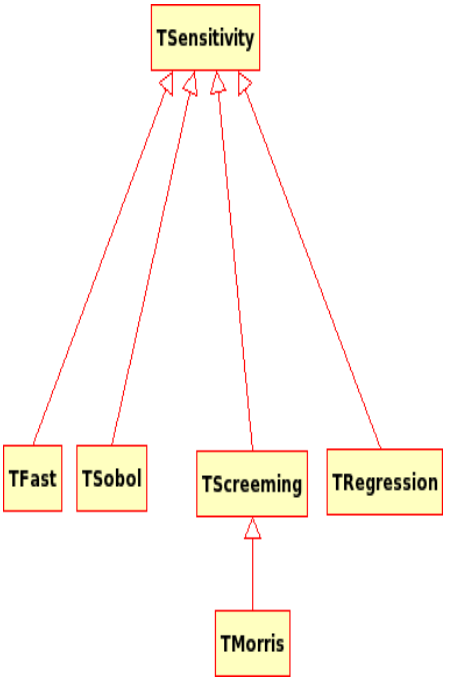




"Sensitivity" library

Application : "Ishigami" function
En cours de développement
Perform a sensitivity analysis between the X and Y matrix

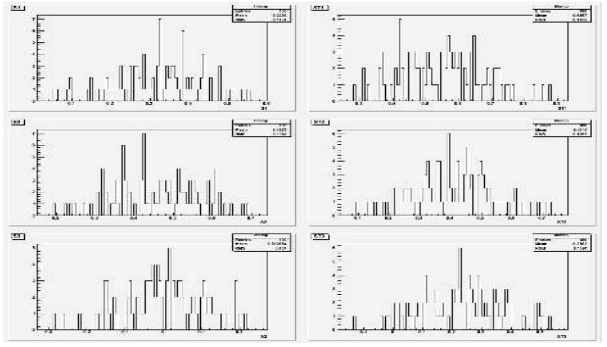
- Regression methods
  - ⋆ Pearson (values)
  - ⋆ Spearman (Rank)
- "Screeming" method as **Morris**
- "Sobol" indexes
  - ⋆ Monte-Carlo
  - ⋆ "Fourier Amplitude Sensitivity Test" **FAST**

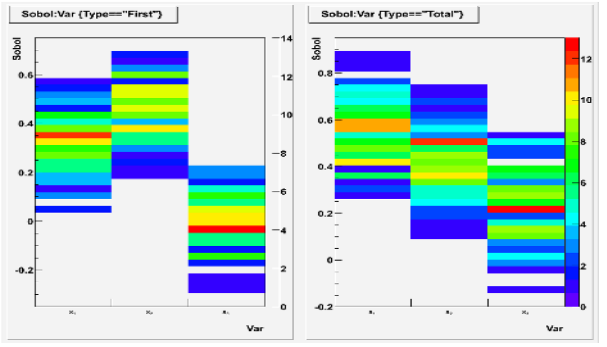


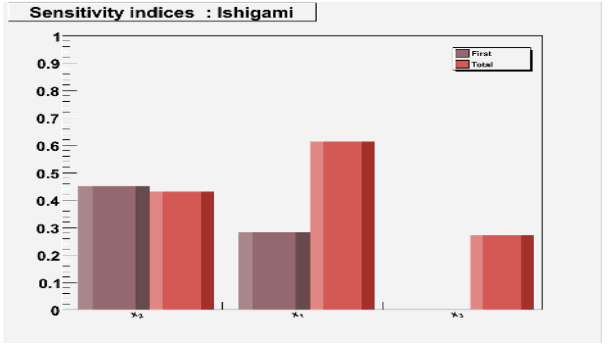
Application : "Ishigami" function

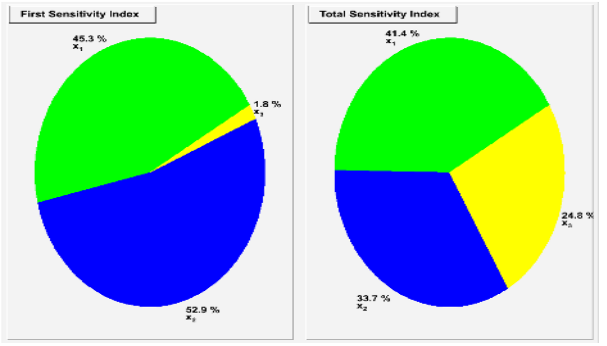
– "Ishigami" Benchmark : $A = 7$, $B = 0.1$, $x_i \sim \mathcal{U}[-\pi, \pi], i = 1, 2, 3$

$$f(x_1, x_2, x_3) = \sin x_1 + A \sin^2 x_2 + B x_3^4 \sin x_1$$









URANIE

"DataServer"

"Sampler" . . .

"Launcher" . .

"Modeler" . . .

"Optimizer" .

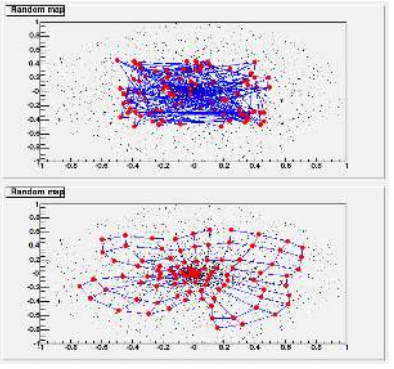
"UncertModel

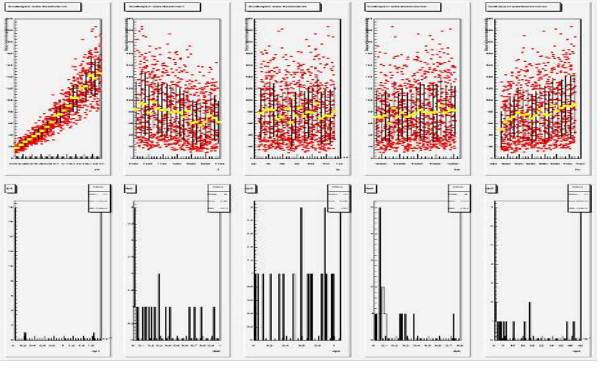
"Sensitivity" .

Plan de . . .

En cours de développement

- Quantification vectorielle
  Carte de Kohonen, Neural Gas,
- Test Statistiques :
  Shapiro-Wilks
  Kolmogorov-Smirnov
  Cramer-von Mises
  Anderson-Darling
- Analyse de sensibilité par les tests :
  Common MeaNs (CMN)
  Common MeDians (CMD)
  Common Locations (CL)
  Common Variances (CV)
  Statistical Independence (SI)
- Manuel Utilisateur (60%)





Plan de développement 2009-2010

- v1.2 12/2008
  - ⋆ Mise en LGPL
  - ⋆ Module NISP, tests (sensibilités)
  - ⋆ Manuel Utilisateur
  - ⋆ Portage sous Windows DataServer + Sampler
- v2.0 06/2009
  - ⋆ Documentation + formation Anglais
  - ⋆ Mise en place d'une MCO, Portage complet sous Windows
  - ⋆ JRC methods RBD (Random Balance Designs) + HFR
- v2.2 12/2009
  - ⋆ Livraison dans le cadre de ROOT
  - ⋆ module FORM/SORM
- v2.4 12/2010
  - ⋆ Mise en place des méthodes RaFu (IRSN)
  - ⋆ Méthodologies intégrées "déterministes/statistiques" (Karlsruhe/Pise)
  - ⋆ ANISP

Formation

Support aux utilisateurs ...